Modified Freeze-TCP for Improved Performance in Mobile Wireless Networks

A thesis
submitted in partial fulfilment
of the requirements for the degree
of
Master of Engineering in Electrical and Electronic Engineering
At the
University of Canterbury, Christchurch, New Zealand.

by

Gustavo M. T. Da Costa

Supervisor
Associate Professor Harsha R. Sirisena

Department of Electrical and Computer Engineering
University of Canterbury
Christchurch, New Zealand

April 2002

Table of Contents

Table of Figures

Tables

Chapter 1. Introduction

1.1. Background and Motivation

With the rapid development of wireless network technologies, such as cellular networks, the demand for wireless data access is becoming a reality. Third-generation (3G) technologies are set to enter the market in the very near future, allowing increased bandwidth and marking a shift from the circuit-switch mode characteristics of first and second-generation cellular networks to packet mode operation. Future wireless networks (such as a future 4G cellular system) are expected therefore to move towards an all-IP network, so that wireless networks become a part of the global Internet. Such an all-IP network has the potential to provide cost savings through simplified integration among different networks and enhanced service capabilities to a wide base of consumers.

Given the wide deployment of the Transmission Control Protocol (TCP) as the Transport Layer protocol in wired networks and the desire to allow internetworking between wired and wireless networks with minimal intervention from intermediate nodes, so that information can be accessed at any time without incurring major changes in existing infrastructure or disruption in performance, it makes sense to use TCP not only in the wired Internet but also in the wireless network.

However, TCP is based on assumptions that are not valid in wireless environments, which leads to poor performance in wired-wireless networks. TCP performance problems exist not only due to the wireless channel characteristics but also due to mobility in cellular-based networks, as the mobile host (MH) hands off from one cell coverage area to the next. A considerable effort has been made in recent years to modify TCP for wireless and mobile environments.

The protocol behaviour dependence on the particular packet loss characteristics (e.g. short duration vs long duration) and frequency of losses encountered makes the solutions studied to be targeted to the characteristics of a particular environment. Furthermore, the underlying network characteristics call for different approaches to be used depending on the level of support that can be obtained from the components of the network and its characteristics.

Some works have shown that the TCP performance degradation is particularly serious in cases where a connection is lost for extended periods. Although a broader discussion of TCP performance problems and solutions is presented, it is on this specific type of problem that we focus our attention in this thesis.

1.2. Research Objectives

The objectives of this thesis are:

- Survey the causes for TCP performance degradation in wireless environments.
- Survey proposed improvements to TCP in a wired-wireless environment.
- Investigate key requirements for efficient TCP mobility support. This involves not only TCP modifications but also consideration of the network infrastructure.
- Develop a scheme that allows improved TCP performance but does not require changes to the fixed end-host.
- Investigate through simulations the performance achieved by the proposed scheme and how it compares with other proposals.
- Discuss the strengths and weaknesses of the proposed scheme.

Although we survey and discuss problems related to high losses in the wireless channel, we focus the development of the proposed modifications towards improving the TCP performance in the presence of handoffs. However, consideration is given to the needs of reducing performance degradation due to the wireless environment so that future work can be performed to extend the proposed scheme.

1.3. Thesis Overview

This section provides an overview of the thesis structure and briefly states the main purpose of each chapter.

Chapter 2 presents an overview of the Transmission Control Protocol (TCP) and the reasons why it performs poorly in wireless and mobile environments. This leads to a discussion on the main aspects that need to be observed when attempting to modify TCP for such environments. This includes a discussion not only on TCP aspects but also on network support and architecture aspects that influence the performance achieved by TCP.

Chapter 3 presents a survey on TCP modifications for wireless environments proposed in the literature with a discussion of strengths and weaknesses based on aspects noted in Chapter 2. Lessons learned from this survey are summarised and conclusions drawn at the end of the chapter.

Chapter 4 describes the environment in which we seek to improve TCP and provides a description of two proposed extensions to one of the approaches surveyed in the literature, Freeze-TCP, which is particularly suited to our target environment and objectives. Consideration is also given to practical aspects of this mechanism.

One of the extensions proposed aims at making the Freeze-TCP approach more robust to performance degradation caused by packet losses, by avoiding the retransmission timeout (RTO) value inflation in a condition that may occur in some cases after a reconnection when losses occurred.

The other extension presented to Freeze-TCP is a proposal for extending the original Freeze-TCP scheme to avoid the need for end-to-end disconnection prediction signalling through the use of buffering at the wired network while still avoiding the need to snoop the TCP header. Its applicability and alternative options for the

buffering mechanism and its interaction with other wired network components are also considered.

Chapter 5 presents an overview of the simulator used and its limitations as well as a more complete discussion of the implementation of the schemes discussed in Chapter 4. Modifications made to the original simulator are also included in this chapter. Besides, this chapter gives the methodology and performance metrics for the simulation studies conducted in Chapter 6.

Chapter 6 presents a simulation study of the main issues involved in the approach focused on in this thesis, namely using the persist mode feature of TCP flow control to avoid TCP performance degradation during disconnections. As opposed to previous studies, consideration is also given to the interaction of the approach considered with unmodified TCP so that potential interference with unmodified TCP can be highlighted.

Although the performance achieved is dependent on the particular environment and configuration used, the aim of this chapter is to present a simulation study of the schemes implemented in the simulator during the course of the research so as to provide a better understanding of performance issues relating to the approaches studied and highlight their strengths and weaknesses.

Chapter 7 concludes the thesis by highlighting the main observations made throughout it and the strengths and weaknesses of the proposal made in Chapter 4, based on an analysis of the results presented in Chapter 6. Suggestions for future work to further develop the methods studied in this thesis are also presented in this chapter. It is hoped that this discussion will provide directions to the continuation of the work in this thesis based on the lessons learned, particularly in the sense of a future practical implementation in a testbed for detailed performance studies in a real environment.

Chapter 2. TCP in Wireless Environments

2.1. TCP Overview

This section presents the main characteristics of the Transmission Control Protocol (TCP). The emphasis is on providing a brief overview of how the protocol works and its main characteristics related to our work. For a detailed study of the protocol the reader is referred to [Stevens 1994] and [Comer 1991].

2.1.1. Protocol Basics

TCP is a connection-oriented protocol that provides a full-duplex, reliable service to the application layer. A TCP connection is uniquely identified in an internet by its Socket Pair (<Sender IP Address, Sender Port Number; Receiver IP Address, Receiver Port Number>). It achieves reliability by requiring that the receiver acknowledge the correct reception of data to the sender. The Checksum header field allows the receiver to detect segments received in error. Such segments are not acknowledged and therefore are discarded so that the sender will hence need to retransmit them. The basic TCP header is 20 bytes long, if no options are present. The use of a 4-bit Header Length field limits the header length to a maximum possible size of 60 bytes.

The basic unit of transfer passed from TCP to IP is called a segment. The protocol uses cumulative acknowledgements to detect the correct reception of data at the receiver[1]. If the receiving TCP gets a segment with sequence number higher than the next in order sequence number expected, it sends a Duplicate Acknowledgement

[1] Unless the Selective Acknowledgement (SACK) option is used.

(DUPACK) with the next in order byte it expects to receive instead of acknowledging the new data[2].

The receiving end TCP advertises how much data it is willing to accept from the sending end TCP, based on the amount of available buffer space at the receiver. This flow control avoids the receiving host running out of buffer space due to a faster sending host.

2.1.1.1. Congestion Avoidance

Congestion is avoided by using a sliding window mechanism [Stevens 1994]. The sender hence maintains a congestion window (CWND), which is in units of segments, that grows as new acknowledgements are received, since such ACKs imply that data has successfully left the network. The sender transmission window is hence the minimum of the advertised window and the CWND. The sliding window mechanism therefore requires the sender to know which packets have been acknowledged, the packets that have been sent but not acknowledged and the number of packets that are allowed to be sent. This is depicted in Figure 2.1.1.1-1 [Goff et al 2000]. As data sent is acknowledged, the left edge of the window moves to the right. Similarly, as more data is allowed to be sent, as determined by the CWND and the advertised window, the right edge of the transmit window slides to the right, allowing more data to be sent.

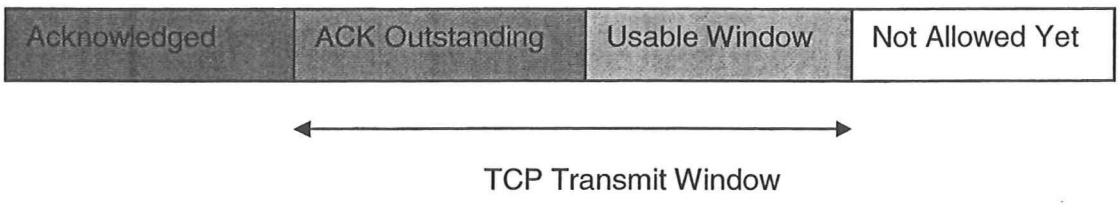

Figure 2.1.1.1-1: The TCP Transmission Window.

[2] As opposed to an ACK acknowledging new data, a DUPACK cannot be delayed by the receiver.

In order to avoid congestion, the congestion window has an initial value of one segment and grows exponentially in the slow start phase by increasing the congestion window value by one and sending two new segments each time a new ACK is received. This cycle ends when the congestion window reaches the value of the slow start threshold (SSTHRESH), which is initiated to a high value. When SSTHRESH is reached, the congestion avoidance phase is then entered, where the congestion window grows linearly with one segment per RTT until a loss is detected.

2.1.1.2. Loss Detection and Recovery

TCP assumes that losses are due to congestion and hence on detecting a loss it reduces its congestion window so that the congestion can be reduced. As acknowledgments are received again the congestion window restarts to grow. SSTHRESH is halved on detection of a loss.

A lost segment is detected by the sender TCP if it does not receive an ACK before the expiration of a timer or if a threshold number of consecutive duplicate acknowledgements (DUPACKs) is received. To maintain a timer value that reflects the current level of network congestion, round trip time (RTT) measurements are taken by the sender. RTT measurements are done by incrementing a counter for every "tick" of a TCP timer routine. The interval between ticks is the timer granularity and the minimum round trip timeout (RTO) is two ticks.

The sender is able to calculate the RTT by timing a segment per connection and calculating the RTT when an ACK that acknowledges the timed segment is received based on the current time and the time when the timed segment was sent. If the Timestamp Option defined in RFC 1323 [Jacobson et al 1992] is used, a timestamp is placed in each segment sent reflecting the time the segment was sent and the sender calculates the RTT based on the timestamp value echoed by the receiver. This option adds 12 bytes to the standard 20 bytes TCP header.

Therefore, the receiver keeps the value of the last acknowledgement sequence number sent (last_ack_) and the value of the current timestamp to echo (ts_echo_). When a segment with a timestamp option that advances the left edge of the receive window and contains the byte in last_ack is received, the timestamp value in the segment is saved and used as the timestamp value to echo. This value is hence used in the timestamp echo reply field of the timestamp option.

[Wright and Stevens 1995] describe a situation in which the loss of an ACK can cause incorrect inflation of the RTT estimate. In such case, the algorithm of RFC 1323 will not update ts_echo_ for the retransmitted packet sent after a timeout following the loss of the ACK. Hence the sender will calculate the RTT upon receiving the ACK for the retransmission using the timestamp of the original segment instead of the timestamp of the retransmitted segment. Therefore, the sender takes into account not only the time elapsed between the retransmission and the reception of the ACK for the retransmission but also the time elapsed between the original transmission and the timeout.

The algorithm presented in [Wright and Stevens 1995] to correct this problem does not require the left edge of the receive window to be moved in order to update ts_echo_. It updates the current timestamp to echo variable ts_ echo_ if the timestamp in the segment received is greater than or equal to ts_echo_ and the sequence number in the received segment is less than or equal to the next in-order segment expected by the receiver. This is referred to as the timestamp echo bug fix in this thesis and is present when using the timestamp option in our simulations.

The retransmission timeout value is based on the smoothed RTT (SRTT), which is an estimation of the average RTT, and on the RTT variance (RTT_VAR). TCP does not use the RTT samples of retransmitted segments to update the retransmission timer. This is known as Karn's Algorithm and avoids the retransmission ambiguity problem, where the sender cannot know if the ACK received after a retransmission is for the original segment or for the retransmitted segment. Using such an RTT sample could lead to either unnecessarily inflated RTO or to underestimation of the RTO, which could lead to spurious timeouts. The RTO value is then given by

$$RTO = SRTT + 4 \times RTT\_VAR$$

Where,

$$SRTT = \alpha \times RTT + (1-\alpha) \times SRTT$$

and

$$RTT\_VAR = \beta \times |RTT - SRTT| + RTT\_VAR \times (1-\beta)$$

α = Gain of average = 0.125

β = Gain of deviation = 0.25

Since TCP assumes that losses are due to congestion, it reduces its congestion window to the initial value in case the timer expires and starts the slow start phase. Hence, a timeout is intended to indicate a persistent congestion.

On the other hand, if the sender is still receiving DUPACKs and the DUPACK threshold is reached, this means that the congestion level is not as severe as in the case where no ACKs are received, since in the former case packets are still leaving the network. The fast retransmit and fast recovery algorithms are then used.

Therefore instead of reducing the congestion window to the initial value and starting slow start, TCP sets SSTHRESH to half the current CWND, retransmits the missing segment and sets the CWND to SSTHRESH + 3. Since data is still flowing in the network, the sender increases its CWND each time it receives another DUPACK and transmits a new segment, if permitted by the new CWND size. This cycle ends when an ACK that acknowledges new data is received. The CWND is then set to SSTHRESH and the sender is then in congestion avoidance. This mechanism avoids the need for relying on a timeout to detect congestion, which is difficult due to the possibility of rapid changes in the path RTT[3].

[3] This is especially true when wireless environments are involved due to the fast variation of the wireless path characteristics.

2.1.2. TCP Persist Mode

This section briefly describes the specifications in RFC 1122 [Braden 1989] relating to the actions taken by the sender when a zero window advertisement (ZWA) is received.

RFC 1122 states that the advertised window should not cause the transmission window to be shrunk so that the usable window does not become negative. However, for robustness of the protocol, the RFC specifies that the sender must be able to cope with this situation. It is recommended that outstanding data in the ACK Outstanding region of the new transmission window (Figure 2.1.1.1-1) should be retransmitted. Additionally, outstanding data that falls within the Usable Window region of the new transmission window may also be retransmitted. The RFC specifies that in any case the lack of acknowledgement from data beyond the right edge of the new window should not cause the connection to timeout.

In case the receiver takes longer to process and remove data from its receive buffer than the rate in which it receives data, the receiver will eventually run out of buffer space and send a ZWA to stop the sender from sending more data. Upon receiving a ZWA, the sender freezes its retransmission timers and the congestion window. The sender leaves persist mode when an ACK with a non-ZWA is received. This is sent by the receiver when buffer space becomes available at it.

Since the sender has stopped data transmission, it must be capable of restarting transmission later while avoiding deadlocks if the ACKs sent by the receiver specifying an advertised window larger than zero are lost. To cope with this situation TCP implementations must have a probing mechanism to take the sender out of the persist mode.

Therefore, the TCP sender in persist mode periodically sends a Zero Window Probe (ZWP) to the receiver in order to see if the receiver is still advertising a zero window. As with the retransmission timer, the interval between these probes is exponentially backed off. The sender must allow the connection to stay open as long as the receiver

acknowledges the ZWPs. Although a ZWP should contain one new byte of data, many TCP implementations, such as Linux and FreeBSD, do not include any data in the ZWPs [Goff et al 2000]. We consider this situation, where the ZWPs do not contain any data, in our simulations.

It is recommended that the first ZWP be sent after the persist mode has existed for the duration of the retransmission timeout period. Although RFC 1122 recommends that the first ZWP should be sent when the persist mode has existed for the duration of the retransmission timeout, many implementations use a lower bound of five seconds for the persist timer [Stevens 1994]. The interval between probes grows exponentially until it reaches a certain value. Many implementations use a value of 60s for the maximum interval between ZWPs [Stevens 1994].

RFC 1122 points out that the sender may ignore a window update with a smaller window than previously advertised if the acknowledgement containing the reduced advertised window does not acknowledge new data.

2.2. Causes of TCP Performance Degradation in a Wireless Environment

The fundamental cause of TCP performance degradation in a wireless environment is its assumption that losses are due to congestion. This action is inappropriate in a wireless environment since packets are often lost in the wireless link due to corruption, temporary fading or mobility [Cáceres and Iftode 1994]. Commonly stated bit error rates (BER) in wireless channels are in the order of 10^{-6} or even worse, if local retransmissions and forward error correction (FEC) are not used, as opposed to 10^{-12} in wired networks [Cáceres and Iftode 1994]. The use of local retransmissions, however, can cause inflation of the round trip time if the link layer attempts in-order packet delivery. This can lead to spurious timeouts [Ludwig and Katz 2000].

The reduction of the congestion window each time the sender times out or there is a fast retransmission due to the number of duplicate packets reaching a threshold[4] causes the sender to under utilise the available resources.

After a loss the congestion window grows according to the slow start and congestion avoidance phases. This means that a considerable amount of time is taken for the sender transmission rate to reach the previous level if the congestion window is unnecessarily reduced. Furthermore, this time increases with the RTT of the path since the increase in the congestion window is dependent on the reception of acknowledgements.

Host mobility, on the other hand, incurs not only the problem of reduced congestion window and the consequent time to grow it, but it can also lead to longer disconnection times while the sender is unable to receive packets. This can lead to serial timeouts and significant idle times even after the mobile has regained connectivity [Cáceres and Iftode 1994].

In handoff scenarios, it may be prudent for the sender to reduce the congestion window following a handoff to avoid overloading the new cell if the new cell is significantly more congested than the previous one. In most cases, however, the new cell will have similar congestion characteristics to the old cell's and hence the reduction in sender transmission rate is unnecessary.

2.3. Summary of Important Aspects to Improve TCP Performance

Important factors that should be observed when looking at improving the performance of TCP for wireless environments are:

- If the congestion window is reduced due to link error or temporary mobile disconnection, it should be able to increase the congestion window back fast to

[4] Three DUPACKs in most implementations

the appropriate level for the current link congestion state.

- Avoid timeouts and reduction of congestion window when a packet is lost due to error or if a temporary disconnection or handoff occurs. However, in the case of a handoff, it is also important to consider the traffic conditions at the new coverage area, as using the same window may not be suitable given the new conditions. This can lead to congestion and hence packet losses.
- Avoidance of successive timeouts is especially important since the retransmit timer is backed off for each unsuccessful retransmission attempt (up to 64s in many implementations [Stevens 1994], which leads to large inactive periods after connection is regained.
- Stop transmission of data from the sender while the MH is disconnected to avoid wasted bandwidth at the wired network, and so that the gateway and base stations are not overloaded with new data while no data can leave the gateway.
- Local retransmissions of data lost due to disconnections so that end-to-end retransmissions are not required and recovery is made faster. An important consideration in this case is the avoidance of the problem of competing retransmissions at the sender and the BS [Ludwig 2000].
- Restart data transfer as soon as the connection is re-established to avoid unnecessary idle periods.
- Minimise the effect of RTT variations measured by the sender to avoid spurious retransmissions or spurious timeouts [Ludwig and Katz 2000].

2.4. Important Network and Mobile Host Aspects for Effective TCP Mobility and Wireless Support

Effective TCP support in wireless/mobile environments is not simply dependent on changes in the TCP protocol. The characteristics of the network, such as its architecture, will play a fundamental role in determining the amount of loss seen at the transport layer and hence will be directly related to the performance degradation observed.

Network Requirements

- Minimise Packet Losses

The network architecture and its capabilities dictate the amount of data lost during a handoff. Methods that can be used to accomplish this task include redirection of packets after a handoff [Perkins and Johnson 2001] and buffering of packets in anticipation to a handoff using multicast [Seshan et al 1996]. However, consideration to the type of traffic must be given. Therefore real-time traffic, which is delay sensitive, may trade-off increased packet losses for reduced delay while less delay sensitive applications (e.g. FTP) trade-off increased delay for a reduction in packet losses.

The use of hierarchical network architectures [Cáceres and Padmanabhan 1998] and recent micro-mobility protocols [Gustafsson et al 2001], [Campbell et al 2000] reduce the amount of time required to complete a handoff when compared to Mobile IP and hence have an important role in the amount of performance degradation seen by TCP. The micro-mobility protocol proposed in [de Silva 2001] uses a hierarchical micro-mobility architecture where the micro-mobility routing protocol explicitly signals the Gateway FA of an incoming handoff through the current BS before the MH loses connection with its current BS. Hence the Gateway FA can start buffering, thereby avoiding losses during the handoff completion time. The reception of a registration request from the new BS indicates a handoff occurred and hence the buffered packets are sent to the MH through the new BS.

- Deploy ability

The ability of interoperation with existing infrastructure is a primary concern. Therefore, TCP modifications should ideally be restricted to the MH. In particular, changes to the fixed hosts should not be required due to the large number of existing fixed hosts, making impractical their modification to support mobile hosts. It is also

important, from a practical point of view, that schemes do not require modifications at both ends of the connections in order to function. If changes at both ends are required, the scheme must be capable of operating as unmodified TCP in case the other end of the connection does not support the changes required by the new scheme. Hence the use of the enhancements must be negotiated at connection set-up time [Mathis et al 1996].

However, such schemes would still lead to poor performance in the observed throughput for a considerable time into the future, as deployment of such schemes in fixed hosts would not become a reality for a long time. Furthermore, a considerable variation in performance would be observed by the mobile terminals, depending on whether or not the fixed host has the required enhancements.

The possibility of packets and acknowledgments taking different paths from sender to receiver and vice versa is another point of concern since this could cause problems if base stations were involved in the traffic control and recovery process.

- Scalability and Efficiency

Approaches that require large amounts of processing at the base station or any other point in the network have the problem of excessive complexity and the risk of performance degradation especially as the traffic volume increases.

- Security Issues

As highlighted in [Goff et al 2000], the use of IP payload encryption, such as IPSec [Kent and Atkinson 1998], makes it impossible for a base station that is not part of the security association to snoop a packet and do the mediation required if a scheme requires identification of values in the TCP header.

Mobile Host Considerations

Since mobiles use scarce battery resources, the amount of computation required by the MH is also a concern. Hence a trade-off exists between system scalability due to complexity of network components, such as base stations, and use of limited mobile node resources. The limited processing capability of mobiles also make it important to attempt to reduce the amount of transmissions and receptions made by the mobile host.

2.5. Summary

An overview of relevant aspects that lead to TCP performance degradation in wireless environments was presented in this chapter. Important considerations that should be observed in attempting to adapt TCP for wireless environments, not only at the TCP itself but also related to the underlying network, were discussed.

It was emphasised that the main cause for TCP performance degradation is its assumption that packet losses are due to congestion and the inappropriate actions taken in wireless environments due to this assumption. Therefore, the central problem of TCP is its lack of differentiation of congestion detection and control from error detection and recovery.

It was also emphasised that the network characteristics play a fundamental role in the level of performance degradation seen by the TCP sender, as the network configuration can aid in hiding losses from TCP, in the fast recovery of lost packets and in packet loss avoidance.

Chapter 3. Survey of Approaches to Improve TCP Performance over Wireless Networks

3.1. Introduction

Now that the main causes for the TCP performance degradation in wireless and mobile environments as well as the major trade-offs and considerations in modifying TCP have been discussed, a survey of various proposed modifications to TCP in order to improve its performance in mobile wireless environments is presented. Some remarks about the strengths and weaknesses of these proposals, based on experimental research reported in the literature, are also presented.

The chapter concludes with a summary of the main features to be observed in a modified TCP and some important lessons learned from previous studies. This summary is a basis for the development and evaluation of the modifications proposed in this thesis. A more comprehensive survey of TCP modifications for wireless and mobile environments can be found in [Vaidya 1999].

3.2. Error Recovery Schemes

For reviewing the work targeted mainly to wireless error recovery, the classification presented in [Balakrishnan et al 1997] is often used. This classifies the schemes in split connection, link-layer or end-to-end approaches.

3.2.1. Split Connection Approaches

Split connection approaches such as I-TCP [Bakre and Badrinah 1995] and MTCP intend to shield the FH from errors introduced by the wireless environment, which

significantly reduce the TCP performance [Duchamp and Reynolds 1992], by using a separate connection over the wireless link. Since the mobile host is usually one hop away from the BS, this approach takes advantage of the small round trip time (RTT) over the wireless link, as in link-layer approaches, to provide fast recovery of the congestion window in case of wireless related window reductions. Therefore this approach tries to provide faster adaptation to the highly dynamic characteristics of the wireless link due to factors such as high BER or fading errors while also avoiding the need to modify the existing fixed host TCP implementation.

Split connection approaches can also take advantage of the fact that, by using a connection for the wired link and another for the wireless link, a protocol optimised for the wireless link can be used. A performance enhancement proxy (PEP) is used to do the mediation between the two connections and ensure the correct reception of packets by the MH. Figure 3.2.1-1 summarises the split connection approach [Ludwig 2000].

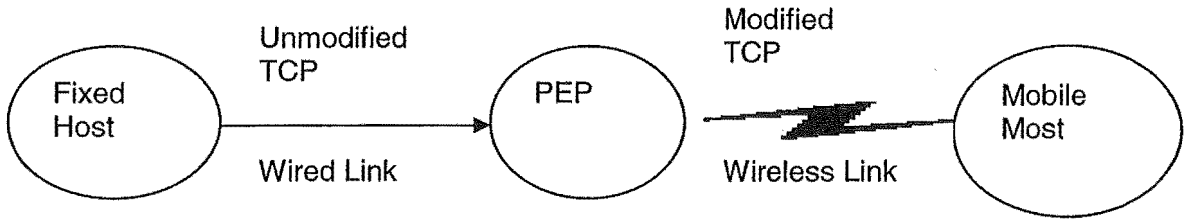

Figure 3.2.1-1: The Split Connection Approach.

The next paragraphs outline the main problems with this approach and some observations from previous works.

Problems with the Split Connection Approach

The main and most frequently cited argument against the split connection approach used in I-TCP is the fact that the gateway between the wired and wireless networks can acknowledge packets even before the MH receives the data packet. This means that the end-to-end TCP semantics is violated.

Since the wireless connection successfully transfers data at a slower rate than the wired connection, there is a tendency for packets to accumulate at the gateway, leading to excessive buffer requirements at the BS. In this scheme, during a disconnection at the wireless connection, the FH will continue to receive acknowledgements from the BS and hence will continue sending data at the rate determined by the wired link conditions. Therefore, a long disconnection could easily lead to buffer overflow if no measures are taken to prevent this problem. I-TCP uses the TCP advertised window flow control at the wired connection to avoid buffer overflow at the BS. Also, the presence of a separate connection over the wireless link leads to increased processing delays and complexity at the gateway (BS), which requires processing up to the transport layer rather than only up to the network layer.

[Balakrishnan et al 1997] found that a split connection using TCP-Reno over the wireless link performs almost as badly as the original end-to-end TCP-Reno. Timeouts at the BS cause the sender to stall due to BS buffer overflow. Furthermore, the authors found that the split connection method performs much better if SACK is used in the wireless connection (assuming no packet reordering). Therefore, they conclude that the main gains obtained by split-connection approaches come from TCP optimisations, such as the use of SACK, rather than the splitting of the connection. [Brown and Singh 1997] found that a split connection approach using unmodified TCP over both connections also performs poorly in the presence of lengthy disconnections. Therefore, it has been shown that it is not essential to split the TCP connection at the BS to improve performance and hence the problems previously mentioned due to the connection split can be avoided.

The problems with the split connection approach discussed above have changed the focus of more recent research towards the development of end-to-end schemes that can improve the TCP performance and still retain TCP's end-to-end semantics [Montenegro et al 2000]. Hence the end-to-end approach relies on more involvement of the connection end points, rather than concentrating changes at the BS.

3.2.2. Link Layer Proposals

As in the split connection approach, the link-layer proposals also take advantage of the small round trip time (RTT) over the wireless link to provide fast recovery of errors in the wireless link. This can be done because in most cases the mobile host is only one hop away from the BS. Therefore this approach tries to provide faster adaptation to the highly dynamic characteristics of the wireless link due to factors such as high BER or fading errors by providing local recovery through link layer retransmissions at the wireless link. Figure 3.2.2-1 summarises the link layer retransmissions approach.

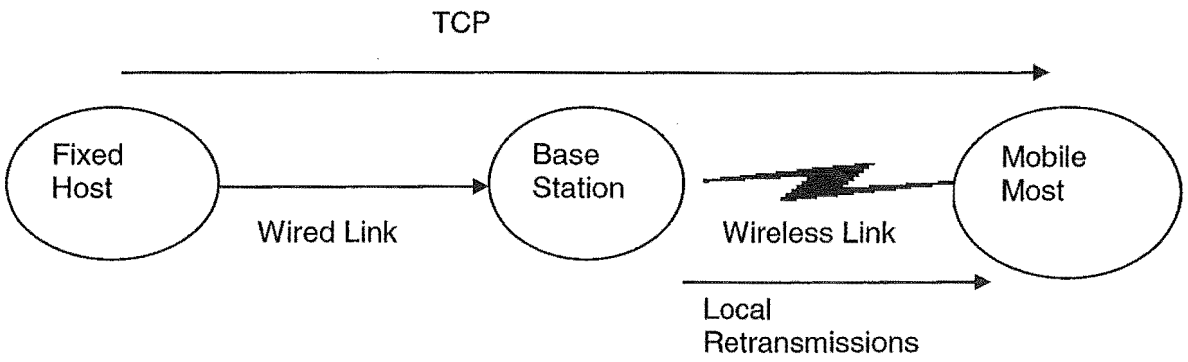

Figure 3.2.2-1: End-to-End Approach with local recovery.

Local recovery is generally possible if the TCP timer at the source has a coarse granularity, such as the commonly used 500ms timer granularity of many current TCP implementations. The link layer protocol uses a finer granularity so that the link layer has enough time to perform several local retransmissions before the TCP retransmission is invoked. Hence, this is a method of avoiding the need for end-to-end retransmissions while still maintaining the end-to-end TCP semantics. The link layer proposals are usually based on either error correction based on Forward Error Correction (FEC) or on retransmission of lost packets in response to Automatic Repeat Request (ARQ) [Balakrishnan et al 1997].

However, since this type of approach is based on retransmission of data at the BS, although it is useful in dealing with high BER and short disconnections, it is not very useful for handoffs or long disconnection period situations. This is because in such

cases no ACKs are received at the BS and hence a timeout may occur at the sender, reducing the congestion window to its initial value and thereby reducing the throughput considerably. Furthermore, since the MH is unable to receive data, the retransmissions will simply waste precious wireless bandwidth.

Competing retransmissions at the sender and at the link layer can also cause the invocation of congestion control, which leads to performance degradation, as was concluded in [DeSimone et al 1993]. This would be a more serious problem if the TCP sender uses a finer timer granularity, as has been proposed in [Floyd 1994] so that TCP can react faster to signs of network congestion.

[Balakrishnan et al 1997] also showed that link layer retransmissions may lead to low throughput and goodput due to TCP fast retransmissions caused by the link layer delivery of out of order packets to the MH and the consequent generation of duplicate acknowledgements. The authors have reported results that show that 90% of the lost packets were retransmitted by both the source and the base station. [Ludwig 2000] studies this problem of link layer and TCP interference in detail and presents a modification to the TCP sender to avoid the problem of spurious fast retransmissions when out of order packets are delivered by the link layer as well as the problem of spurious timeouts, due to increased RTT, that may occur if the link layer attempts in-order delivery.

3.2.2.1. TCP-Aware Link Layer Protocol

One of the most widely referenced proposals to deal with wireless channel impairments is the Berkeley Snoop Module (Snoop) proposed by [Balakrishnan et al 1995]. This protocol caches data received from the FH and sends the data to the MH. If a loss is detected, through the arrival of duplicate acknowledgements at the BS or the expiration of the local timer, and if the BS has the lost data cached, it retransmits the data and suppresses the DUPACKs. This avoids the sender invoking congestion control due to DUPACKs and also avoids the need for time consuming end-to-end retransmissions. Therefore, this method is often referred to as a link-layer TCP aware scheme, even though it is implemented at the IP layer. As with other link-layer

proposals, a coarse granularity timer is needed at the TCP source while the Snoop agent uses a finer granularity timer for this scheme to be effective.

This method does not use two separate connections and maintains end-to-end semantics. Several studies have shown that this scheme provides significantly increased throughput and goodput in the presence of wireless errors, e.g., [Balakrishnan et al 1997], [Chen and Lee 2000].

The main problem with this method is that, as with other methods that depend on an intermediate node involvement, if the IP payload is encrypted, the snoop module at the BS cannot inspect the TCP header to determine if the packet lost is in its cache. Also, this scheme is not effective when long disconnection periods exist, as the sender will still timeout and the performance will suffer.

In the experiments presented in [Balakrishnan et al 1997], it was found that the best link layer scheme was composed of Snoop with Selective Acknowledgements. This provides efficient local retransmissions based on selective acknowledgements, while also shielding the sender from DUPACKs caused by wireless link errors using the Snoop agent, which prevents the invocation of fast retransmission.

3.2.2.2. Delayed Duplicate Acknowledgements

The generation of DUPACKs due to the loss of packets in the wireless link or the delivery of out of order packets by the link layer mechanism and the consequent performance degradation described previously led [Vaidya et al 1999] to develop a TCP-unaware scheme based on the Snoop protocol. The idea of this scheme is to perform timer based local retransmissions and avoid sending duplicate acknowledgements without the need to involve the BS in DUPACK suppression. This has the advantage of reducing the BS complexity and especially it allows the use of encrypted IP payloads, since now the BS does not need to snoop the header.

This is possible by using link-layer acknowledgements to trigger retransmissions at the BS instead of TCP ACKs. Unnecessary fast retransmissions are avoided by

holding the third and subsequent DUPACKs at the MH for a maximum fixed amount of time to allow for local retransmissions to send the missing data.

[Vaidya et al 1999] observed that, as with Snoop, this scheme is particularly useful for large bandwidth-delay product networks, where enough packets to trigger a fast retransmit may be in transit. By contrast, if the congestion window cannot frequently grow sufficiently to trigger a fast retransmit, there is not much gain in suppressing DUPACKs. It was also observed that the performance of this scheme depends on the relative frequency of losses due to transmission errors compared to those losses due to congestion since holding DUPACKs may delay congestion action at the sender. The choice of a proper fixed delay in the face of a constantly changing RTT is difficult and is an area requiring further research.

Since during a handoff, or time of disconnection, no DUPACKs can be sent back to the BS and packets locally retransmitted will not successfully reach the mobile, this type of approach is irrelevant in dealing with handoffs and mobile disconnections.

3.2.3. End-to-End Approaches

As opposed to the ideal network approaches previously described, end-to-end approaches follow the ideal sender approach where the sender performs the desired action based on information from the receiver, an intermediate node in the network, or a combination of both. Although our objective is to avoid changes to the TCP sender at the fixed end-host, we briefly review some main proposals that require changes at the TCP sender as they provided guidance to some possibilities for our work. These schemes may also be viable in the longer term. SACK is not discussed here since it is presented when the choice of a baseline TCP for our study is discussed (Section 5.2).

Negative Acknowledgements (NACKs) have been proposed in [Chan et al 1997] to reopen the congestion window after a reconnection and so allow faster retransmission of lost data at the TCP sender. These NACKs are also used to trigger probe packets that request the available bandwidth information estimated by the receiver based on the elapsed time between probe packet arrivals so that proper values for SSTHRESH

and CWND are estimated and congestion due to excessive CWND can be avoided. SACK principles are also used in this scheme.

The Explicit Bad State Notification (EBSN) scheme [Bakshi et al 1997] attempts to avoid timeouts at the fixed host source during local recovery by the base station. Therefore, the approach used in this scheme is for the base station to send an EBSN message after it is unable to deliver data to the mobile host. This causes the previous timeout to be cancelled and a new timeout value identical to the previous one is then used. If no new acknowledgements or EBSNs arrive, a timeout will occur and the congestion control will be performed as usual.

The authors of the EBSN scheme have shown that EBSN provided significant improvements over TCP-Tahoe over LANs (up to 50%) and WANs (up to 100%) in their simulation study.

[Balakrishnan et al 1997] describe an Explicit Loss Notification (ELN) method where the sender avoids reducing the congestion window when ACKs marked by the receiver to indicate that a packet was lost in the wireless link are received. The authors of ELN also propose to combine ELN with SACK to avoid timeouts due to more than one loss in a window of data.

The Eifel Algorithm [Ludwig and Katz 2000] prevents spurious retransmissions by modifying the TCP sender to detect a spurious timeout or fast retransmit and then avoids the reduction of the SSTHRESH and CWND. It uses timestamps in the TCP header to be able to extract the information required to eliminate the retransmission ambiguity problem.

3.3. Handoffs and Disconnection Recovery Schemes

TCP performance degradation can be avoided by using overlapping cells, so that the handoff happens before the mobile loses connectivity with the serving cell and the new BS may be notified before the connection to the old BS is lost [Cáceres and Padmanabhan 1998]. However, this may become infeasible if small cells are used

[Cáceres and Iftode 1994]. Furthermore, temporary disconnections due to fading or lack of bandwidth may lead to timeouts at the TCP sender, which can only be avoided through the modification of the TCP protocol. Frequent beaconing is also useful in speeding the completion of the handoff, however this consumes bandwidth even if no handoff is in progress [Brewer et al 1998]. Handoff losses can also be avoided by tunnelling packets from the old to the new BS. This avoids losses if the cells do not overlap enough to allow the completion of the handoff before connection to the old BS is lost, e.g., [Cáceres and Padmanabhan 1998].

The basic Snoop scheme was modified in [Seshan 1995] so that when a BS predicts a handoff or a handoff is requested by the MH, the nearby BSs join a multicast group. The cache at the current BS is then sent to the BSs in this multicast group. Hence, when a MH moves into a new BS, this BS already has the state information and no expensive state transfer is required. This avoids the need for the new BS to build a new cache, during which period the performance would be similar to standard TCP since the BS would not be able to perform local retransmissions for previously lost packets and congestion control would be invoked.

The problem with this approach is the larger redundant traffic volume introduced by the transmission of packets to the BSs to which the MH will not move. Hence this could also present problems in high traffic situations due to the increased buffering needs at the BSs and increased load on the network with redundant multicast.

The idea to use fast retransmissions to speed the recovery of the congestion window after a disconnection occurs was first proposed in [Cáceres and Iftode 1994]. In this approach, instead of the sender waiting for a timeout after the disconnection has elapsed, lower layers will signal a reconnection to the mobile. The mobile can then send three duplicate acknowledgements to the sender, which will cause an immediate fast retransmission. The authors show that this method can provide significant improvement in the time taken to resume communication after a handoff is completed, since it avoids the need to wait for a potentially long RTO.

The problem with this scheme is that, if several packets are lost in a window of data, the sender may need to recover through a timeout and hence the congestion window is still reduced all the way to its initial value. The bug fix proposed in [Floyd 1994] and used in many TCP implementations also does not allow for more than one fast retransmission in a window of data.

The most important schemes related to the work in this thesis are summarised next.

M-TCP

M-TCP [Brown and Singh 1997] attempts to improve TCP performance in environments with frequent or lengthy disconnection and variable, low, bandwidth. It is a hierarchical, split connection based, scheme that uses TCP persist mode (Section 2.1.2) to avoid timeouts and congestion window reduction due to a prolonged disconnection. It attempts to maintain end-to-end TCP semantics by not sending an acknowledgement from the gateway to the FH until the gateway receives the acknowledgement back from the MH.

The authors point out that smaller cells will be required in the future so that a larger bandwidth can be made available to MHs. The architecture used in this scheme groups several Mobile Support Stations (BSs) under the control of one Supervisory Host (SH). This avoids the need to have state transfer if the mobile moves between cells covered by BSs controlled by the same SH and the handoff can be greatly simplified and faster when compared to the case of I-TCP. It also avoids the need to modify a large number of BSs, hence the BSs can be kept simpler. State transfer is only required if the MH moves into a BS controlled by a new SH. This structure is hence appropriate for high mobility and small cell situations.

The SH performs a number of functions, such as bandwidth management, local recovery and monitoring of DUPACKs. The wireless connection in M-TCP can also perform data compression to increase the efficiency of the limited bandwidth wireless connection and no congestion action needs to be taken by the wireless TCP connection due to the control performed by the bandwidth management module. In

order to enable retransmissions from the SH after a reconnection is detected, packets are cached at the SH so that it can perform retransmissions if a reconnection is detected.

This method uses a timer at the gateway (SH) to monitor the flow of ACKs coming from the MH so that zero window advertisements can be sent to the FH when a possible disconnection is detected at the SH. This avoids the sender wasting bandwidth at the wired network and from timing out, which would invoke congestion control measures leading to reduced data throughput.

The SH needs to hold the last ACK received by the MH so that if a disconnection is detected, by the expiration of the SH timer, the SH can have a new ACK in which it can advertise the zero window and pass this information to the sender to ensure that the ZWA is not ignored by the sender (Section 2.1.2). This will then avoid the invocation of congestion control.

Since in M-TCP the SH can dynamically allocate bandwidth, the SH can also send a zero window advertisement if it detects that the cell where the MH is does not have enough bandwidth.

A greeting packet is sent to the SH once the MH regains connection so that the SH can send the last ACK held at the SH advertising a non-zero window size to the sender and transmission can be re-initiated as soon as the connection is re-established.

The problem with this approach is the high complexity at the SH which has to control a number of BSs and MHs. Hence scalability is compromised. Since a timer-based approach is used, it is also necessary that the sender use a coarser granularity to avoid the problem of competing retransmissions [Balakrishnan et al 1997] due to local retransmissions.

A similar scheme to M-TCP, although simplified, was proposed in [Chan et al 1997]. It uses the same approach of putting the sender in the fixed network in persist mode with the aid of intermediate nodes in the fixed network, except that no SH is used.

Hence, the BS is responsible for handling all the handoff requirements and the requirements to enter and exit persist mode with zero window. Buffering is used to store the packets during handoff. The BS maintains a local timer that upon expiration allows it to set the advertised receiver window to zero in the last ACK, which is held by the BS as in M-TCP. As opposed to M-TCP, this scheme does not use a bandwidth estimation module that can be used to set the advertised window according to the estimated wireless link capacity.

The TCP-SMART proposal [Elaoud and Ramanathan 2000] also uses an agent at the BS to buffer packets and retransmit them locally. DUPACK filtering is performed at the BS and the BS places the FH sender in persist mode by sending a DUPACK with a ZWA. Therefore the last ACK does not need to be held at the BS as in M-TCP but the ZWA being a DUPACK can be ignored by the sender, according to the observation in M-TCP that a new ACK should be used to place the FH in persist mode.

Freeze-TCP

Instead of reacting to a disconnection after it has happened, [Goff et al 2000] proposes a proactive scheme (Freeze-TCP) to allow the MH to inform the TCP sender in the fixed network of a possible disconnection. The aim of this approach is to avoid packet losses and congestion control action by the sender during periods of disconnection or handoff without requiring modifications to the intermediate nodes or the fixed sender.

It makes the receiver send one or more zero window advertisements to the sender when the receiver senses an impending disconnection. It is suggested that the disconnection prediction can be made based on the measurement of received signal strength. Hence the scheme tries to avoid losses by taking action before a disconnection occurs rather than reacting to losses.

The scheme also uses triplicate ACK (TR-ACK) of the last acknowledged packet as in [Cáceres and Iftode 1994] when the connection is re-established, so that the sender can leave the Persist Mode faster rather than waiting for the next scheduled time to

send a ZWP to test if the receiver is still disconnected. This is important because the sender exponentially backs off the time between ZWPs [Braden 1989]. Furthermore, although RFC 1122 recommends that the first ZWP should be sent when the zero window has existed for the duration of the RTO, many implementations use a minimum ZWP time of 5.0 seconds [Stevens 1994]. Hence the wait for the next ZWP could be greater than the disconnection period itself.

The idea of Freeze-TCP is to send a zero window advertisement (ZWA) only when the power level is low and in a way that the ZWA can be sent one RTT before the signal is actually lost. This allows the sender to stop before the disconnection actually happens and packets in transit still arrive at the MH before the disconnection happens. Hence, sending the ZWA one RTT before the disconnection avoids the early placement of the sender in persist mode, which would lead to unnecessary sender idle time before the disconnection happens. If the sender is stopped too late, packet losses will occur and congestion control action will still occur.

The authors of this scheme showed improved performance for the scheme under various scenarios in the case where the RTT was known and the warning period was set to this value during the tests reported. As expected, the gain is greater when the delay-bandwidth product is larger since the benefits of avoiding the congestion window decreasing are larger than when the window is only allowed to grow to a small value or frequent window drops occur anyway due to other factors such as network congestion.

TCP Advertised Window Control Functions

[Onoe et al 2001] propose a scheme to couple the signal strength based disconnection prediction mechanism, as proposed in Freeze-TCP, with buffering at the BSs to allow a reduction in the amount of packet losses observed during a disconnection.

In this scheme, the MH signal strength measured at the BS is used to start buffering at the BSs. If the signal recovers the BS restart transmission to the MH but if the signal is kept low for a certain amount of time, the BS sends a ZWA to the TCP sender. The

Mobile IP [Perkins 1996] bind-update extension is used to trigger tunnelling of buffered packets to the new BS after a handoff is completed. The advantage of this approach over Freeze-TCP is that the threshold can be made smaller than the threshold required by Freeze-TCP since the delay between the MH and the BS is smaller than the delay between the MH and the FH. Therefore, packets that would have been lost by Freeze-TCP because the sender did not stop in time to avoid the loss of data in-flight in the fixed network still have a chance of being buffered at the BS and sent to the correct BS after the handoff. The BS sending a ZWA to the sender can be seen as a backup for cases where the MH was unable to predict a disconnection.

On the other hand, the proposal described in [Onoe et al 2001] would require the snooping of the TCP header to send a ZWA from the BS to the correct TCP senders after the MH signal is found to be excessively low. Therefore, this scheme cannot be used when the IP payload is encrypted. Furthermore, this scheme increases the processing requirements at the BSs, which must monitor the signal strengths and store information to send a ZWA when required.

Although [Onoe et al 2001] also consider other methods to avoid losses during a handoff we only consider the TCP Advertised Window Control Functions (which we call TAWCF) described above as it relates to the TCP zero window advertisement procedure studied in this thesis.

Their results show that using persist mode based on the measured signal strength can provide significant improvements over unmodified TCP. In the scenario simulated in their paper, a throughput improvement of approximately 79% was observed for the MH advertised window control implementation while adding the BS buffering mechanism provided approximately 160% improvement.

3.4. Summary

In this chapter the main approaches presented in the literature to modify the current TCP for better performance in wireless environments were presented. The main advantages and drawbacks of the schemes were also highlighted. It was seen that

schemes usually are targeted to either the problems of high loss in the wireless link due to corruption or to the problem of long disconnections caused by long fades or handoffs. It was also seen that current research favours not splitting the TCP connection at intermediate points to avoid violating the end-to-end semantics. Finally it can be seen that a combination of schemes targeting the different problems would be necessary to provide enhanced TCP performance in a wireless environment.

The survey shows that link layer retransmissions are an efficient way to deal with sporadic errors in the wireless link provided that competing retransmissions and excessive RTT inflation are minimized. On the other hand, handoffs and long disconnections are dealt with well by putting TCP into persist mode to avoid timeouts and congestion window reduction while avoiding the need to modify the sender host.

The issues of determining when a disconnection will happen and its implications were explored in greater detail and are presented in Section 4.3. In Chapter 4 we also propose a modification to the buffering scheme triggered by a MH disconnection prediction to avoid the need for snooping the TCP header and still allow the use of intermediate node buffering.

Chapter 4. Enhancement Mechanisms for Freeze-TCP

This chapter first introduces the network environment being considered in our study. Then a study of the idea of sending a ZWA (Zero Window Advertisement) based on the signal strength at the MH (Mobile Host) is presented and its strengths and weaknesses are discussed. This provides the basis for the proposed changes to improve the performance of Freeze-TCP.

4.1. Network Environment Studied

In this thesis, the focus is on the case where a fixed host sends data to a mobile host, which is a common case for bulk data transfer. FTP is used to transfer data to the MH and we are interested in avoiding modifications to the wired sender. This means that the sender does not have direct knowledge of the receiver mobility or wireless link conditions.

The environment of interest is a cellular network where the mobile periodically moves into a new cell, requiring a handoff. It is assumed that a link layer mechanism exists so that the sender is not affected by the corruption losses typical of wireless links. In this way it is possible to isolate and study the problems caused by disconnections to TCP.

We consider a large delay between the sender and receiver in order to investigate the effects on the TCP end-to-end recovery mechanism. In the environment studied, the cells do not overlap. This highlights the problems suffered by TCP when a handoff occurs and connection with the fixed network is temporarily lost. The communication to and from the mobile is considered to go through a base station. No ad-hoc communication between mobiles exists.

We consider a network structure where there is a node that is common to two or more base stations and this point can act as a controlling point to the traffic going to the base stations involved in the handoff. This architecture is consistent with that required for use of the Regional Registration [Gustafsson et al 2001] micro-mobility scheme and also cellular networks where there is a Gateway FA or MSC.

Figure 4.1-1 illustrates the simulated network topology. The topology used is similar to that in [Cáceres and Iftode 1994] except that we also explicitly include background traffic in a number of tests and the sender is not located at the Gateway FA but instead is at a distant point.

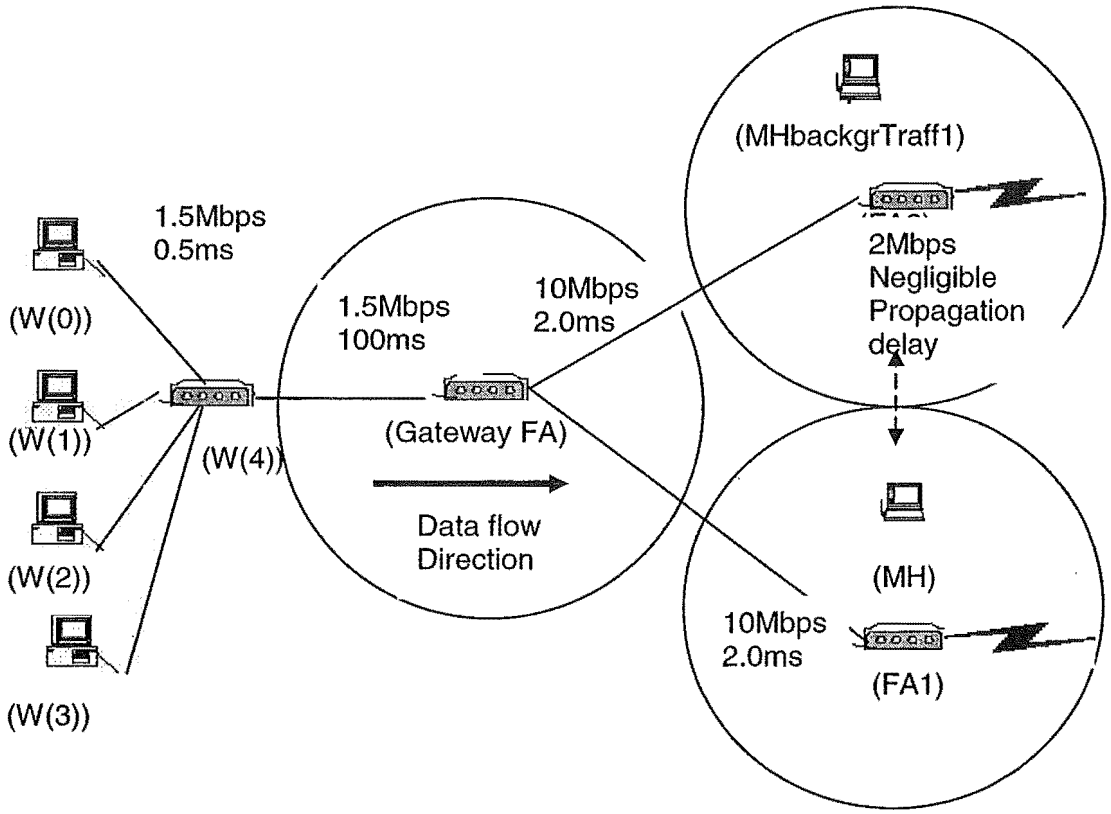

Figure 4.1-1: Network environment for handoff simulation.

The links connecting W(0-3) with the W(4) are 1.5Mbps as is the link between W(4) and the Gateway FA, unless otherwise stated, to avoid link mismatch and congestion losses in the wired network.

4.2. Scheme Overview

The approach used to avoid data losses and timeouts during disconnection was to seek to put the TCP sender into persist mode prior to a disconnection and hence avoid loss of packets so that congestion control is not unnecessarily invoked. This follows the approach suggested in [Goff et al 2000] where the mobile node predicts a disconnection based on the signal level measured at the mobile.

Freeze-TCP is extended in our work to make the MH aware of the disconnection duration using its ability to detect a disconnection and notify this to the Transport layer, together with the ability to detect a reconnection. This allows a correction in the timestamp returned to the TCP sender based on the estimated disconnection duration so that the effects of the disconnection in the RTT calculation can be minimised in case the Timestamps Option is used.

In order to avoid losses and avoid the need for tunnelling packets among base stations, an architecture similar to that used in M-TCP is adopted and buffering to recover from packet losses due to handoffs are performed at the gateway. As in [Onoe et al 2001], two methods are used to reduce the buffering requirements at the gateway: putting the sender into persist mode and using disconnection predictions to trigger buffering only when needed.

Two possible buffer mechanisms are discussed. The first one differs from the work in [Onoe et al 2001] in that, instead of blocking the buffer, the buffer caches the received data if it receives a disconnection prediction and sends the original data. The other mechanism considered is to also block the buffer upon reception of a disconnection prediction from the MH but with a mechanism that allows the use of certain types of encryption. This differs from the work in [Onoe et al 2001] in that their mechanism requires the snooping of TCP headers and hence does not allow the use of IPSec security.

Attention is also given to the issue of avoiding excessive bursts of data following a reconnection to avoid congestion losses.

The following sections give a detailed explanation of the proposed scheme.

4.3. Mobile Host Actions

This section describes the issues involved and the actions performed by the MH in the proactive disconnection detection scheme based on the Freeze-TCP concept considered in this thesis. The particular method to measure the signal strength in a practical implementation is not within the scope of this work. However, we consider how the information that a disconnection is about to happen can be communicated to the TCP layer, so this is an indication of how to perform the cross layer information transfer in a real implementation.

4.3.1. Disconnection Prediction at the MH Network Interface

The MH network interface monitors the received signal strength as in Freeze-TCP. It is possible to decide that a disconnection is about to happen based either on the monitoring of beacons' signal strength or the measured signal strength of the data received. Mobile hosts capable of performing mobile controlled handoffs must be able to monitor the signal strength of the beacons and hence could use this information to decide when to send ZWAs. However, if the mobile is to be able to react to temporary fades, the beacon period used may not be small enough to allow the detection of the disconnection period. On the other hand, it is not desirable to use excessively small beacon periods to avoid increasing the load in the wireless link, where bandwidth is usually scarce and should be conserved. It is therefore preferable to monitor the data signal strength so that even for handoff cases the disconnection prediction is not strongly dependent on the beacon period.

The approach used here is to base the decision to warn the sender of an incoming disconnection on the received data signal strength rather than the beacon signal strength. The drawback of this approach is the increased processing needed at the MH

to monitor the signal strength. Hence, a future practical implementation must study the best way to perform signal strength measurements so that the proper tradeoff between TCP performance and MH processing power requirement can be determined, based on the particular implementation requirements.

4.3.1.1. Warning Threshold Power Choice

Putting the sender into persist mode based on the received signal strength requires an adequate choice of a warning threshold power. As noted in [Goff et al 2000], the ZWA should ideally be sent one RTT before the actual disconnection so that the ZWA can reach the sender and any packets in transit will reach the MH before the disconnection actually happens. If buffering is performed at the network, the warning period required is the RTT between the buffering node and the MH instead of between the sender and the MH. This shows that buffering at an intermediate node close to the MH allows more packets to be sent before the disconnection without incurring losses.

In order to simplify the MH and minimise power usage, the disconnection threshold level is fixed [Goff et al 2000], [Onoe et al 2001] and is related to the minimum threshold power that the received signal must have to receive data without error. This level is MH device specific. This approach was used in the simulation implementation for Freeze-TCP and its enhancements considered in this thesis.

Although a fixed threshold was used on our simulation implementation, a dynamic capability to adjust the threshold could lead to better performance by allowing the threshold to more closely track the RTT of the connection thereby allowing for variations in the RTT during the connection.

The MH can match the warning period to the estimated RTT by attempting to make the elapsed time between the first ZWA sent in response to a disconnection prediction and when the disconnection happens as close as possible to the connection RTT. The MH can estimate the warning period duration by keeping two variables that record the time when the last ZWA was sent and the time when the first ZWA was sent due to

the prediction of a disconnection by the lower layers. This would allow calculation of the estimated warning period as:

Warning_period_ = time_last_ZWA_sent_ - time_first_ZWA_sent_

If the estimated RTT at the MH is larger than the estimated warning period found using the expression above, the threshold used is too low and should be increased to avoid loss of packets. On the other hand, if the estimated RTT is smaller than the estimated warning period, the threshold can be reduced to avoid freezing the sender too soon.

If the MH receives packets until when the disconnection happens, the warning period estimate will be accurate. However, if the MH stops receiving packets sooner than the disconnection, the warning period estimated will be smaller than the actual available warning period. This can cause the threshold to be unnecessarily increased due to the incorrect warning period estimate.

It is possible for the MH to know the connection RTT in cases where a two-way connection is used, that is, the MH also sends TCP data to the fixed host. Hence it measures the RTT in the standard way (see Chapter 2). In a one-way connection, however, the MH would need to send periodic probe packets to estimate the RTT. This has the disadvantage that it wastes wireless bandwidth. Since we consider one-way, FTP, bulk transfer, the development of such mechanism is not undertaken further in this thesis and is left as a suggestion for future enhancement to this work.

The use of a fixed warning threshold power at the MH to reduce its complexity raises the issue of robustness against changing network conditions and specifically the issue of how different warning thresholds will affect the performance. As in the Freeze-TCP case, if a disconnection is not predicted the performance will simply be that of unmodified TCP [Goff et al 2000]. Therefore, an issue to be studied in Chapter 6 is how the scheme reacts to a warning threshold that is higher or lower than the required threshold for timely sender freezing.

We therefore wish to make this scheme more robust against failures to timely predict disconnections. It is also interesting to have a scheme that can stop the sender as late as possible without losing data due to disconnection while keeping the MH simple. This requires the collaboration of intermediate nodes from the network, as in the scheme proposed in [Onoe et al 2001]. These issues of intermediate node involvement will be considered in more detail in Section 4.6.

4.3.2. Cross-Layer Information Flow

The use of information related to the link conditions at the TCP layer requires that such information obtained at the network interface be passed up the protocol stack to the TCP layer. As noted in [Goff et al 2000], this cross-layer communication among different layers of the protocol stack is a drawback of this approach. It also violates the modularity of the protocol stack. However, the new challenges presented by wireless links, and the need to correctly identify the causes of different problems, justifies this cross-layer information flow. This area of study is therefore receiving increasing attention at the moment [de Silva 2001], [Goff et al 2000], [Brewer et al 1998].

The information about a predicted disconnection may also be useful for an application that does not use TCP. Such an application may use this information to adapt to an impending disconnection, such as reducing the sending rate to conserve battery power, since the prediction indicates that packets are more likely to be lost. Real-time applications could even use this information to stop transmission until a reconnection is detected.

However, since we are interested in the behaviour of TCP, in this thesis only information transfer to the TCP layer is considered. A mechanism that may be used to inform TCP of a predicted disconnection is described in this section. Information flow about a reconnection is equally important and is also considered here.

An overview of proposed cross-layer information flow methods is presented in Section 4.3.2.2 and is summarised in figures 4.3.2.2-1 and 4.3.2.2-2.

4.3.2.1. Disconnection Predicted Message Generation and Flow

One possibility to pass the disconnection prediction from the network interface to the TCP layer is to mark an unused flag in the TCP header so that, when this information arrives at the TCP layer, the flag is interpreted and TCP acts as required. This method, however, would require the network interface to check each received packet to see if it is destined to TCP, which may cause excessive power consumption at the MH. Furthermore, acting on the TCP header at the network interface would not be possible if the IP payload is encrypted since the decryption happens at a higher layer (IP). Therefore, an alternative method is considered next for practical implementations. For simplicity of implementation in our simulator, the method of marking incoming packets was used.

A new message that is propagated up the stack to TCP is required. A method that uses aspects of the reactive disconnection detection method presented in [Bakre 1996] to achieve proactive disconnection detection is described next; it could be used in a future practical implementation of our work.

An application process [Bakre 1996] is required to inform other applications of a predicted disconnection. We however only consider here the mechanism to be used to inform TCP of an impending disconnection.

At MH power-up, the application process sends a message to the network interface through TCP as its transport layer. This message information is saved at the network interface. This mechanism allows the network interface to know how to fill the TCP/IP headers because the information is generated as the message flows from the application to the network interface. Upon detecting an incoming disconnection, based on signal strength measurements, the network interface responds to this message and sends it back to the application. The payload of the message informs the application that a disconnection has been predicted.

TCP knows the port number corresponding to the disconnection warning application process and hence passes the message to the application. Upon receiving a message from the IP layer to this port number, TCP sees a bit set in the TCP payload of the disconnection predicted message and knows that it must acknowledge future received TCP packets with a ZWA until a reconnection detection is signalled. This requires the maintenance of a state at TCP.

The modification to inform upper layers of a predicted disconnection is made entirely at the network interface and information is generated only when needed to minimise cross-layer traffic at the mobile. A state is kept at the network interface to allow this. This state is set when the disconnection is predicted and reset when a reconnection is detected, thus avoiding the need to send disconnection predictions repeatedly up the stack. The state can be refreshed periodically at the network interface according to the link conditions.

4.3.2.2. Reconnection Detected Information

A reconnection can be detected by monitoring the reception of beacons and/or registration replies following the detection of a disconnection [Bakre 1996]. [Onoe et al 2001] monitor the beacon signal strength periodically following a disconnection and a reconnection is detected only when the received beacon signal strength is found to exceed a specific threshold.

In our case, since we focus our simulation study in the case of handoffs, a reconnection is detected by the reception of a registration reply following a disconnection. Section 5.1.4 will provide more details about the implementation for such detection mechanism.

Upon reconnection detection, the IP layer, which receives the registration reply packet after handoff completion and passes data packets to TCP, sends a special ICMP message, such as the one defined by the I-TCP protocol, to the application process using TCP as its transport layer. [Bakre 1996] defines a new ICMP message type (ICMP_LINKSTAT) for the locally generated ICMP message. Since this

message is restricted to the MH it does not pose a compatibility problem with other network nodes that do not add the new message type to the ICMP protocol. The ICMP_LINKSTAT _RECONN code used by [Bakre 1996] can be used here also.

The same ICMP_LINKSTAT_RECONN message is sent to the network interface causing the network interface disconnection state to be reset.

Figure 4.3.2.2-1 summarises the cross-layer information flow and the actions taken at each layer in response to the messages received when ICMP message is used to inform MH layers of a reconnection detection.

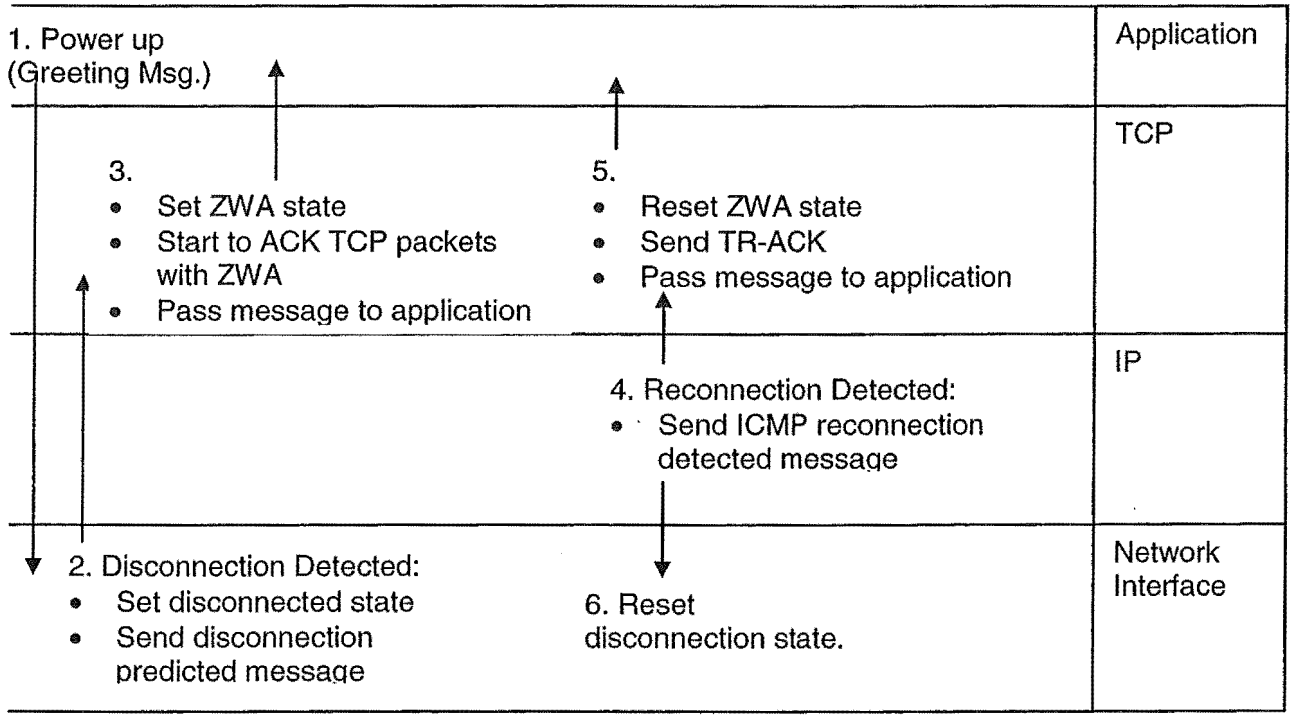

Figure 4.3.2.2-1: Disconnection and reconnection message flow mechanisms using ICMP message to inform of a reconnection.

In order to avoid having to modify the Network layer too, the reconnection detection information can also be sent from the network interface layer if such layer is made capable of monitoring the control channel to detect the reception of a registration reply when a disconnection has been predicted. This indicates the completion of the handoff. The reception of a beacon while a disconnection predicted exists also

indicates that the MH can unfreeze the sender. The same message type sent up the stack to warn higher layers that a disconnection is predicted can be used to inform TCP and applications of the reconnection but the payload information in the message now means that a reconnection was detected.

Detecting a reconnection at the network interface has the advantage of avoiding changes to the MH IP layer. Since the network interface maintains a Disconnection Predicted state set when the disconnection is predicted, the state can be reset there without the need for communication from the Network layer informing the network interface that a reconnection was detected. The message flow and layer actions on receiving the disconnection and reconnection messages are shown in Figure 4.3.2.2-2.

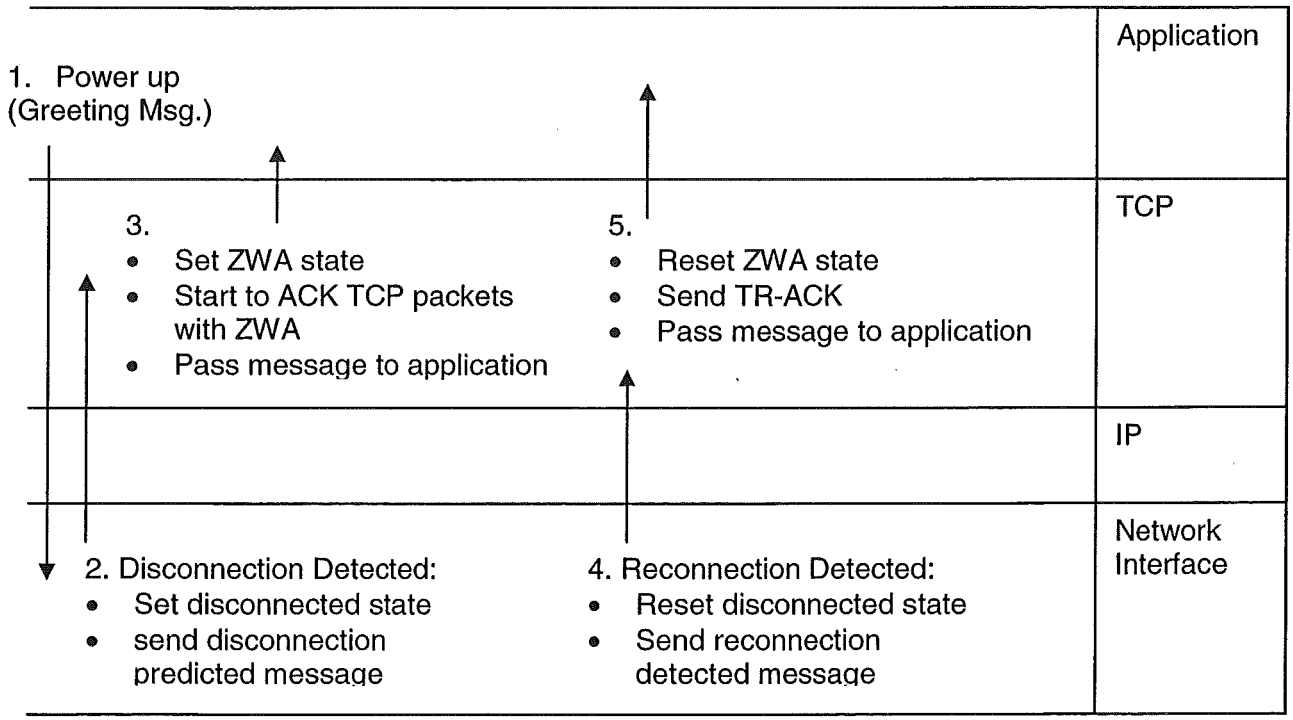

Figure 4.3.2.2-2: Disconnection and reconnection message flow mechanisms.

TCP on receiving the reconnection-detected message will reset the ZWA state. The reception of the reconnection detection message will also cause TCP to send three copies of the ACK (TR-ACK) for the last data packet it received, as in [Cáceres and

Iftode 1994], [Goff et al 2000] so that it is not necessary to wait for the next ZWP from the sender.

The TCP state is reset if a ZWP is received while the receive buffer space is more than zero or if a reconnection detected message is received by TCP. These conditions mean that there is no need to advertise a zero window and the link is capable of receiving data.

Since the ACK with a ZWA is sent prior to the disconnection, as in Freeze-TCP, it is not necessary to hold the ACK for the last byte, as opposed to M-TCP where the Gateway FA needs to hold the last ACK to guarantee that it can put the sender in persist mode. In the case of a MH that sends a ZWA prior to a disconnection, if the MH acknowledges the last byte without a ZWA and then a disconnection is predicted when out of order packets are received, the ZWA would be in a DUPACK and hence could be ignored [Braden 1989]. However, since there is a warning period where received packets are acknowledged with a ZWA, it is likely that at least one ACK for the in-flight data will be for new data and this will put the sender into persist mode.

This mechanism contrasts with the reactive disconnection detection method used by schemes implemented in nodes in the wired network, such as M-TCP, where a disconnection is detected after the connection is lost. In the case of M-TCP, since the disconnection is detected after it happened and hence no more ACK is being received, the Gateway FA has to ensure that it will be able to send a non-DUPACK ZWA. Hence the node has to hold the ACK for the last byte received to ensure that it can send a new ACK if it needs to send a ZWA.

4.4. Inflation of the RTO in Handoffs with Losses

The Fast Retransmissions mechanism of sending copies of the last received packet to restart the sender after a disconnection [Cáceres and Iftode 1994] provides obvious benefits in that the sender will not need to wait for the next timeout after the disconnection period is over to restart sending data. Nevertheless a problem was observed during the study of the performance and behaviour of Freeze-TCP with

different threshold powers which may lead to even worse performance than unmodified TCP-SACK. The problem is described in this section and a solution to it is given in the next section.

According to the Fast Retransmissions mechanism in [Cáceres and Iftode 1994], when the MH detects a reconnection, it will send three copies of the ACK (TR-ACK) for the last in order packet received. If the last segment was successfully acknowledged by the MH and the ACK was received by the sender, these copies of the last ACK sent by the MH will be seen as DUPACKs by the MH. If the number of DUPACKs reach the fast retransmission threshold (usually three), a fast retransmission will take place. However, the last ACK sent by the MH prior to a disconnection can be lost by the MH receiving a data segment prior to a disconnection but sending the ACK when it is already disconnected (if it left the cell coverage area or if fading occurs). Such last ACK sent by the MH can also be lost either if the ACK is corrupted in the wireless link or is lost in the reverse link due to congestion in the wired network.

As seen in Section 2.1, the retransmission timeout value is updated whenever an ACK that was not retransmitted and acknowledges new data (and the timed segment if timestamps are not used) is received. Therefore, if the ACK for the last packet received by the MH prior to the disconnection is lost, the first ACK that was sent to warn of the reconnection and is successfully received by the sender will acknowledge new data. This will cause the RTO to be updated. Since the timestamp echoed in the TR-ACK is for a packet sent prior to the disconnection, the RTT sample used in the new RTO estimate will include the disconnection duration period, which would inflate the RTO estimate.

If multiple packets were lost prior to the disconnection, the MH would need to recover through a timeout. Therefore, the inflated RTO would cause longer idle times at the sender until the next timeout. This situation will be illustrated in Section 6.3 in the context of Freeze-TCP when the MH does not have enough time to avoid the loss of in-flight data and the last ACK sent by the MH prior to the disconnection is lost. Furthermore, since the disconnection is already over, there is no need to include this

time in the current RTO estimate and we would like to have a more accurate estimate of the connection RTT for congestion recovery purposes.

4.5. Modification to Freeze-TCP for Avoiding Excessive RTT Estimates on Reconnection

This section proposes an enhancement to the Freeze-TCP scheme [Goff et al 2000] to avoid the RTO inflation problem after a reconnection is determined, without requiring modifications to the fixed sender. This enhancement is useful in cases where the timestamp option is being used and does not affect the connection if this option is not in use. An improvement to the scheme is also suggested to avoid the need to use timestamps, and may be implemented in the future.

The receiver is assumed to use the timestamp echo bug fix described in [Wright and Stevens 1995] (see Section 2.1.1.2) to decide which timestamp to echo when deciding the timestamp value that is echoed to the sender. A new variable (time_last_ZWA_sent_) is introduced at the receiver that records the time the last zero window advertisement was sent. It is started with the value zero and every time the receiver sends a ZWA in response to a predicted disconnection, the current time is recorded in this variable.

In this way, when the MH determines that the disconnection is over and sends copies of the last ACK with an advertised window larger than zero to restart the sender (as in Freeze-TCP), the timestamp echoed to the sender is modified as:

ts_echo_ = ts_of_last_packet_ + (current_time_ - time_last_ZWA_sent_)

In this expression ts_of_last_packet_ is the timestamp value in the last segment received before the disconnection. Hence the RTT calculation will not include the time the MH was disconnected due to the handoff because (current_time_ - time_last_ZWA_sent_) is an estimate of the disconnection duration and the time stamp echoed is inflated by this time. Since the disconnection is already over, this time should not be included in the RTO estimation because it does not correspond to

the current state of the link. Therefore, the sender (if using the time stamp option) will have a better estimate of the proper timeout value after the disconnection.

Note also that the receiver and sender clocks do not need to be synchronised since the time added to the sender timestamp is the difference of two times measured at the receiver. If delayed acknowledgements are used, the MH should not delay an ACK predicting a disconnection so that the chance of the sender not being frozen prior to the disconnection is minimised. Hence, after a disconnection is predicted the MH does not delay acknowledging segments received so that more ACKs are sent with a ZWA. This provides more robustness against the case where the sender is not frozen due to the loss of ACK with ZWA. In this thesis, however, we do not consider the use of delayed acknowledgements.

We only consider the case where all packets are timed by the sender and the echo sent back by the receiver because we would like to avoid the need to change the fixed host. However, the sender could be modified so that, even if it does not use the timestamp option, it recognises the timestamp echo. Upon reception of an ACK containing a timestamp echo the TCP sender would then use the received timestamp echo, instead of the time when the timed segment was sent, when updating the RTO.

Our updated timestamp can therefore be sent when a reconnection is detected and the sender benefits from it without the need to use the timestamp option. This would be a reasonable approach for implementation in new fixed hosts. Old fixed hosts would have to use the timestamp option in order to make use of the timestamp update mechanism.

4.6. Gateway Buffering Extension to Freeze-TCP

The scheme proposed in [Onoe et al 2001] has the disadvantage that it must be implemented in all base stations and requires the support of bind-update extensions. It also requires the snooping of the TCP header at the BS to allow the TCP sender to be placed in persist mode by the BS. Therefore this scheme cannot be used if the IP payload is encrypted and the BS is not part of the security association. This scheme

also cannot be used if data and acknowledgements go through different BSs. In our environment, however, we only consider the case where the MH communicates with a single BS at a time and so this constraint does not apply.

In order to take advantage of a low warning threshold power and try to make the performance more independent of the threshold choice, we would like to use the approach of buffering at an intermediate node whenever possible. However, we use the network architecture of Figure 4.1-1 to avoid the need to deploy the buffering mechanism in every BS, thus avoiding the need for tunnelling buffered packets between BSs and the need to support bind update messages. The placement of buffering at the Gateway FA instead of at the BS avoids the problem of out of order packets reported by [Onoe et al 2001] and the need to place a sequencer at each BS. We therefore place the buffer agent at the Gateway FA.

The scheme proposed to accomplish these goals is described in the next sections. Differences and simplifications made in the simulation implementation are noted and described in Section 5.1.4. In a real implementation the buffering agent would be located in the IP layer, as in the Berkeley Snoop Protocol [Balakrishnan et al 1995].

Chapter 6 will provide a simulation study of the performance achieved with the proposed scheme. This performance will be compared to our implementation of the Freeze-TCP scheme, from which the scheme described here builds. A comparison will also be made with the base TCP.

4.6.1. Extension Applicability

It is important to note that the buffering is only performed for TCP packets. As discussed in Section 2.4, real time traffic (which normally uses UDP as the transport layer protocol) is sensitive to delays and hence should not be buffered. Therefore, the Protocol field in the IP header is checked and buffering is only performed if TCP is indicated in the Protocol field (value = 6). If the IP payload is not encrypted, it is also possible to check port numbers to see if the packet belongs to an interactive connection and hence should also not be buffered. Otherwise there is a weakness in

that encrypted interactive traffic will also be buffered while it would be better not to buffer these packets.

Although the use of IPSec [Kent and Atkinson 1998] may be a problem for the implementation of agents in intermediate nodes [Ludwig 2000], [Goff et al 2000], it is still possible to snoop the IP header transport field and IP flags and options if transport mode[1] is used in the security model with Encapsulating Security Protocol (ESP) as the security protocol[2]. In this case, the IP header and IP options are not encrypted [Kent and Atkinson 1998]. Authentication Header (AH) on the other hand also provides security to some IP header fields, including the protocol field, and hence our buffer agent cannot act on packets that use this security protocol.

Tunnel mode of IPSec, on the other hand, would encapsulate the original IP header in a new IP header, which would not allow the identification of the Transport protocol, or any other field in the TCP/IP headers, by the buffer agent. Tunnel mode is used whenever one (or both) of the ends of the security association is a security gateway [Kent and Atkinson 1998]. Hence, the buffer agent can only act on packets protected by IPSec if there is an end-to-end security association and ESP is used as the security protocol.

The Gateway FA will not act on the data received if this is encrypted and the Gateway FA cannot access the required information. The next section describes a new extension to the IP header proposed to avoid the need to snoop the TCP header when detecting a disconnection prediction sent by the MH in an ACK packet. This is therefore useful if the IPSec used by the connection is transport mode with ESP as the security protocol. If IPSec is not used, the extension is not needed and the Gateway FA can detect the disconnection prediction through the observation of the advertised window size. If another security mechanism, other than the above, is used this option is not effective and hence should not be used.

[1] Transport mode is typically used for end-to-end communication between two hosts, such as in client-server communication [Stallings 2000].
[2] For an overview of IPSec and network security the reader is referred to [Stallings 2000].

4.6.2. New IP Option to Avoid TCP Header Snooping

One of the objectives of Freeze-TCP is to avoid snooping the TCP header so that the IP payload can be encrypted. Therefore a mechanism that can have the cooperation of an intermediate node for buffering purposes and still avoid having to snoop the TCP header would be desirable. This can be accomplished by using the IP header to trigger the buffering rather than having to look at the advertised window. In IPv4, the reserved bit (bit 0) in the IP flags of the IP header could be used for this purpose. It is set at the MH if required as described below and reset at the Gateway FA so that it can be used at the wired network if required. If IPv6 is used, it would be necessary to include the flag in a header extension. We concentrate on the case of IPv4 in this thesis.

The IP layer at the MH can look at the TCP advertised window of packets to be sent into the wireless link, before encryption is performed[3]. If it is a ZWA, the IP layer sets the flag on the IP header. When the buffer agent at the intermediate node receives a packet, it checks the IP flag and starts buffering or sends buffered packets according to the IP flag of the received packet.

This flag is also set at the buffer agent in all segments that are buffered by such agent. This is done to avoid the case when the MH is still in the low power warning region after a reconnection but is actually moving away from the cell board. When a disconnection is predicted the MH on receiving a segment with the IP flag set can avoid sending a ZWA in the ACKs for these packets.

In this case the MH does not even need to send a TR-ACK since the ACKs with non-zero advertised window will take the sender out of persist mode. To be more conservative, however, the MH still sends the TR-ACK on receiving the registration reply indicating handoff completion so that it can take the sender out of persist mode even if the ACKs for the buffered packets were lost. The MH could also have sent a ZWA (and hence put the sender in persist mode) but no packets are in flight between

the sender and the Gateway FA. Hence the buffer would be empty and it is better for the MH to send ACKs with non-zero advertised window to ensure that the sender will leave persist mode.

4.6.3. Buffering a Copy of the Packet vs. Buffering the Original Packet

The idea of blocking the transmission of data at the intermediate node when a disconnection is predicted in [Onoe et al 2001] is interesting since this allows data to be sent to other MHs that may not be experiencing a disconnection. Thereby the chances of collisions with data that will probably be sent to a disconnected packet and lost anyway is avoided. This has therefore the potential to raise overall throughput.

However, since in our case the buffer agent is placed in the Gateway FA and this node serves multiple BSs and hence is shared by a potentially large number of connections, we also considered the more aggressive transmission scheme of caching the incoming data in the buffer and forwarding the data so that the agent can perform retransmission in the event of a handoff. This follows the same buffering principle of M-TCP and Snoop and allows the buffer to be freed if transmission is successful and an ACK is received. In such case the data acknowledged by the ACK can be removed from the buffer, avoiding excessive buffer requirements.

In a practical implementation the caching method used in the Snoop protocol should be used. In that scheme, kernel mechanisms are used to avoid the cost of data copying [Balakrishnan et al 1995]. Buffering in our scheme, however, is minimised by the fact that it is only required when a disconnection is predicted. This is in contrast to the Snoop protocol, where caching is required for all unacknowledged packets received by the BS so that local loss recovery can take place. Hence the processing cost in our case is minimised by the proactive mechanism used.

[3] [Kent and Atkinson 1998] specified that the IPSec can be implemented between the native IP and local network drivers (BITS implementation) or can be integrated into the native IP implementation at the hosts.

This observation is important since our modifications are used at the Gateway FA, which will have to handle more traffic than a single BS. Therefore, the processing cost of caching packets when required is an area that needs to be studied when a practical implementation is built.

The problem with sending data even after a disconnection was predicted is a potential waste of bandwidth due to the transmission of data that is highly likely to be lost. This is illustrated in Figure 4.6.3-1.

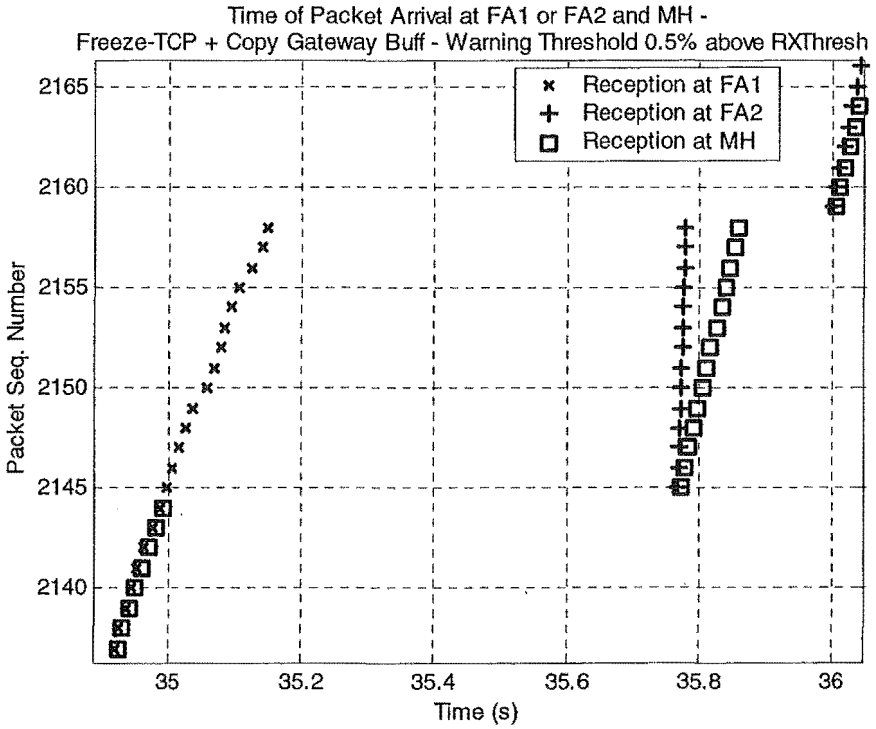

Figure 4.6.3-1: Illustration of packets lost due to handoff and recovery through buffering at Gateway FA.

In the case shown in Figure 4.6.3-1, although packets are lost due to the handoff, the packets are retransmitted by the Gateway FA upon handoff completion.

Figure 4.6.3-2 shows that, although packets were lost (no ACKs received) for the handoff at 35 seconds, the sender does not need retransmissions after the handoff. This is because the Gateway FA retransmitted the lost packets and hence the sender can unfreeze its window and continue sending the data at its previous rate.

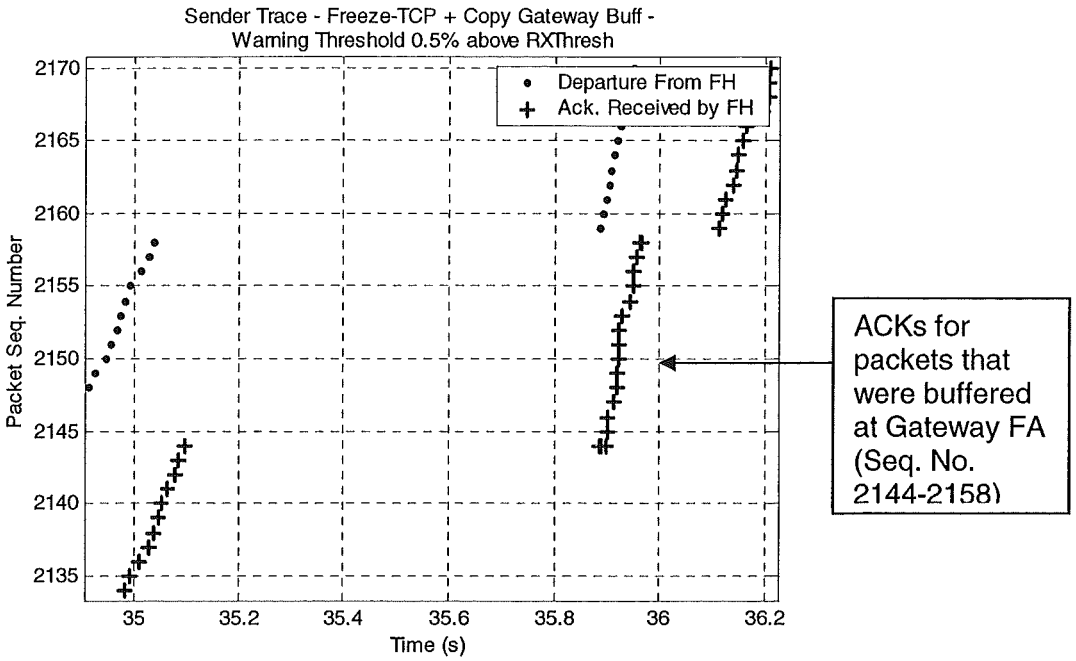

Figure 4.6.3-2: Sender trace after a handoff showing packet losses and recovery at full rate due to Gateway buffer.

This approach does not allow the use of IP payload encryption since the TCP header has to be snooped in order to remove the correct packet from the correct connection upon the reception of an ACK. This proposal also does not allow the use of the BS blocking scheme described by [Onoe et al 2001].

If the Gateway FA keeps data cached for retransmission and the Gateway FA buffer is not blocked but the BS blocks its buffer due to a ZWA, the following situation will occur in case of a ZWA received due to a temporary fade and not a handoff:

At reconnection, the TR-ACK will unblock the BS and the BS will send the buffered data to the MH. When the TR-ACK arrives at the Gateway FA, the buffered packets, which have also been buffered at the BS since the BS is the same as the one prior to the disconnection prediction, will be sent to the MH. This would therefore only create DUPACKs and lead to performance degradation.

If the Gateway FA is blocked upon reception of the disconnection prediction, the Gateway FA buffers the data in-flight between the sender and the Gateway FA. The

BS only buffers data in transit between the Gateway FA and the BS between the time when the disconnection warning is in transit between the BS and the Gateway FA. Therefore no buffer duplication exists and hence no DUPACKs are generated at the MH due to duplicate buffering. Figure 4.6.3-3 summarises the buffer blocking mechanism used. The buffer agent maintains a true/false state that is set when a disconnection prediction message is received and reset when a reconnection warning is received.

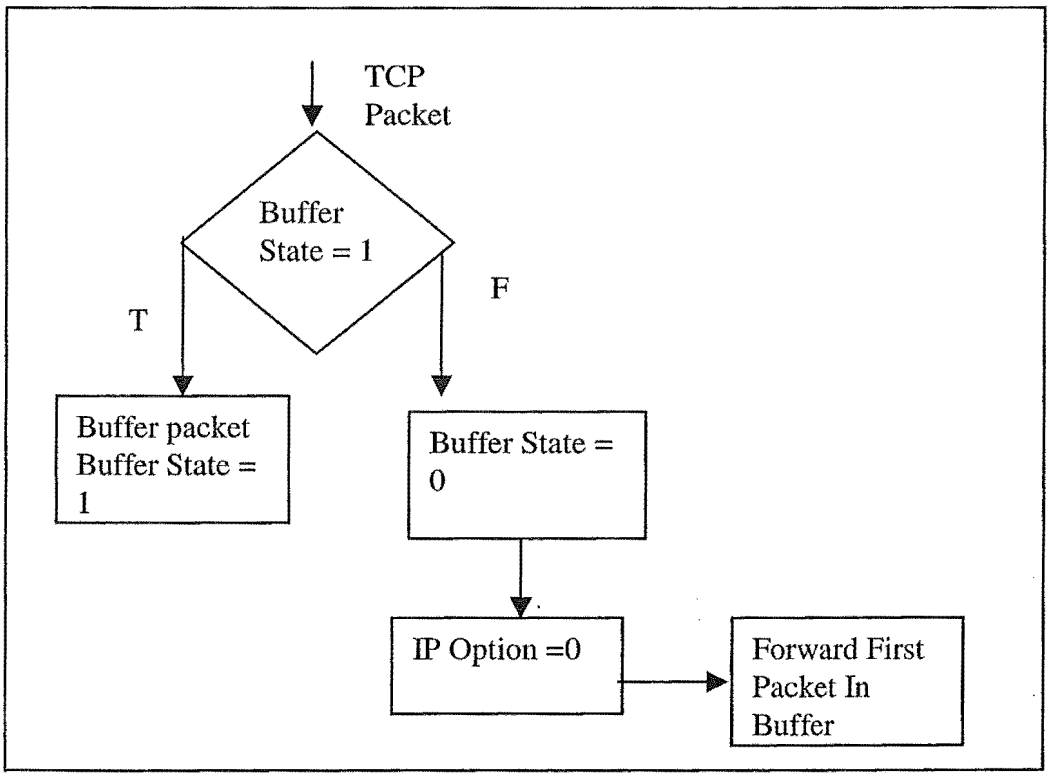

Figure 4.6.3-3: Buffer blocking agent handling of packets to MH.

In the approach of blocking the buffer, if the threshold is set too high, the transmission at the intermediate node will be stopped too soon, wasting valuable time while the link is still capable of sending data. This can be a particularly bad problem if the MH sets a high threshold assuming that not all the infrastructure will support buffering and wants to be conservative ensuring that the ZWA will reach the sender in time to avoid the loss of data in-flight. This is a special concern in the initial stages of deployment since modifications have to be implemented incrementally and hence MHs should

have a higher threshold setting at this stages assuming that it is more likely that network support may not be available.

4.6.4. Buffer Agent Selection

The decision of blocking the buffer until no disconnection is predicted or keeping a copy of the received data for retransmission if required has implications regarding the buffer agent mechanism to be used.

The Snoop protocol [Balakrishnan et al 1995] looks at the TCP header so that buffering is performed per connection, that is, one buffer agent exists per connection. This is required when a copy of the data is kept at the buffer agent so that the agent can discard the correct packet when the data has been acknowledged. If per-connection buffering is not performed and only the source/destination IP addresses are snooped, the ACK could trigger the discarding of a packet from the same source/destination pair but from another TCP connection.

The use of buffer blocking simplifies this mechanism by allowing buffering to be performed per MH.

In the buffer blocking case, when a disconnection is predicted, no more packets are sent so that there will be no duplication of data received at the MH and ACKs received at the buffer agent are for packets that have not been buffered at that node. Hence it is only required to snoop the IP destination address of packets going from the wired network to the wireless network, the IP source address of packets going from the wireless network to the wired network or the MH address in registration requests. Upon reconnection, the buffer agent for the MH can send the buffered data for that MH.

4.6.5. Reconnection Detection at the Gateway FA and Sending Buffered Data

The reconnection detection method used at the MH in our study is through the reception of a registration reply following a handoff or through the reception of a ZWP. This option will be detailed in Section 5.1.4.

At the Gateway FA, two different methods should be used to send buffered packets, depending if the buffer is blocked or if a copy of the data sent is kept at the buffer.

If the buffer is blocked, upon the reception of a reconnection detection buffered packets are sent and the buffer at the Gateway FA is unblocked. This happens if the buffer is in the Buffer state and a packet is received from the wireless network without the new IP Option set, or if a registration request message is received. In our simulator all beacons received by the MH generate a registration request to refresh the binding with the Gateway FA. If the MH does not frequently send a registration request to the Gateway FA, the method used by [Onoe et al 2001] should be used. In this method, after a disconnection is predicted the MH periodically monitors the signal strength of the link, e.g. by monitoring the signal strength of beacons, and sends a reconnection warning to the sender if the signal recovers.

This will avoid the problem of having to wait for a possibly long time until the next registration request sent by the MH. Since the buffer was blocked when the disconnection was predicted, ZWPs will also be blocked at the buffer and hence another mechanism to unblock the buffer must be in place to avoid the deadlock where the sender would stay in persist mode and would not be able to leave it [Braden 1989].

If data is cached and forwarded, the current Care-Of-Address (COA) used by the MH is kept at the buffer agent. If the MH requests a handoff (by sending a registration request with a new COA), the buffered packets are sent, the buffer state is reset and the new COA is kept at the buffer agent. To avoid sending duplicate packets to the MH, the buffer agent examines incoming packets from the wireless network. If an

ACK does not have a ZWA while the buffer state is reset, the buffer agent drops the packets in its buffer having a sequence number smaller than the sequence number in the acknowledgement. This follows the principle of discarding successfully transmitted data used in the Snoop protocol, which also is an IP layer buffer agent mechanism [Balakrishnan et al 1995]. However, Snoop is targeted for packet corruption recovery and hence cannot use the trigger mechanism used in our case. Therefore it has to buffer all received packets.

If buffer blocking at the Gateway FA is used, the use of the link monitoring proposed by [Onoe et al 2001] or the use of registration request reception to indicate proper link conditions as described in this thesis is not essential since the ZWPs are not blocked and hence a deadlock will not occur.

In any case, in order to avoid compatibility problems with existing senders, the IP option should be removed from the ACK packets being sent from the Gateway FA to the wired network.

4.6.5.1. BS Congestion Avoidance Mechanism

Restarting the sender with the previous transmission rate following a MH reconnection is desirable in cases where the MH was disconnected only due to a temporary link failure or if the MH entered a new BS with similar load as the previous one. However, this policy could lead to congestion in cases where the new BS has a higher load than the old one [Vaidya 1999]. Sending all of the packets buffered in preparation for a disconnection from the Gateway FA or from the old BS to the new BS upon the detection of a handoff could similarly lead to losses of these packets. This can happen due to the inability of the new BS to absorb all the packets in the burst.

One possibility to avoid the former problem would be for the MH to advertise a smaller window upon reconnection detection so that the sender will act conservatively. The reception of new segments would then allow the MH to advertise

larger windows to avoid needlessly limiting the transmission rate. The development of dynamic mechanisms based on this principle is an interesting area for future work.

Solving the latter problem of packet losses caused by overload from excessive bursts sent from the old BS in proactive disconnection detection schemes is a more complex issue. In such cases, the new BS sends a request to the old BS for it to tunnel the buffered packets to the new BS and all buffered data is sent to the new BS when this request arrives at the old one [Onoe et al 2001], [Cáceres and Padmanabhan 1998]. If several packets were buffered in advance of the handoff and the new BS has little buffer availability, several of these packets may be lost.

Our scheme of locating the buffer agent for proactive handoff loss prevention at the Gateway FA rather than at each BS is useful in dealing with this problem in a simple way. Therefore, we propose the inclusion of a mechanism to be added at the Gateway FA buffer agent to limit the amount of data that can be sent in a burst.

Hence a limit is defined to the maximum number of packets that can be sent from the buffer agent at any single time the procedure that sends buffered data is called. The buffer agent hence keeps a counter of how many packets have been sent since such procedure was called. This variable is initialised to zero when the procedure is entered and incremented whenever it sends a packet for transmission over the link.

If the configurable maximum burst limit is reached, the function is exited and can only be called again when a registration request is received while the buffer is blocked or if an ACK that does not have a ZWA is received. These indicate that packets have successfully reached the MH and hence more data can be sent towards the MH.

TCP Packets received from the wired network while the Gateway FA buffer agent for the destination MH has buffered data are queued at the Gateway FA's per-MH queue. The packet at the head of the FIFO queue is sent to the MH so that this does not generate out of order packet reception at the MH.

Sending more than one buffered packet for each ACK received helps emptying the queue and minimises buffer size requirements. The TCP ACK clocking mechanism and the TCP receiver flow control mechanism mean that the worst-case scenario for the Gateway FA buffer storage requirement for a TCP connection is bounded by twice the TCP advertised window.

The TCP advertised window is the maximum amount of data that can be unacknowledged at any one time, if allowed by the current CWND size. Hence it is possible that the Gateway FA needs to buffer this amount of data after it received the disconnection prediction from the MH. It is also possible that the TCP sender is allowed to send the entire advertised window worth of data following its unfreezing after reconnection detection. The buffer agent may therefore need to handle the data received prior to the disconnection that has not yet been sent to the MH and the data received from the burst sent by the TCP sender when it was unfrozen.

A future improvement to be considered and added to our current implementation of the buffer blocking scheme with timestamp echo updates and burst limiting mechanisms would be to take into account the Gateway FA congestion level when deciding the burst limit to be used. Hence, if a congestion threshold level is reached at the Gateway FA buffer, the maximum burst size limit would be increased. The study of this mechanism, however, is left for future work and the current implementation in the simulator has a fixed, user defined, maximum burst size. Section 5.1.6 discusses the choice of this parameter in our simulation studies of this mechanism.

4.6.5.2. Buffer Agent Handling of Packets from MH

Figure 4.6.5.2-1 summarises the actions detailed in Section 4.6.5 of the buffer agent actions when receiving data coming from the MH and going towards the fixed network. This refers to how the buffered data is sent to the MH and how the Buffer state is set or reset according to the reception of a registration request or the setting in the IP option flag. The buffer agent sees whether the Buffer State and IP option

satisfies (True) or not (False) the conditions and take the appropriate action shown in the figure.

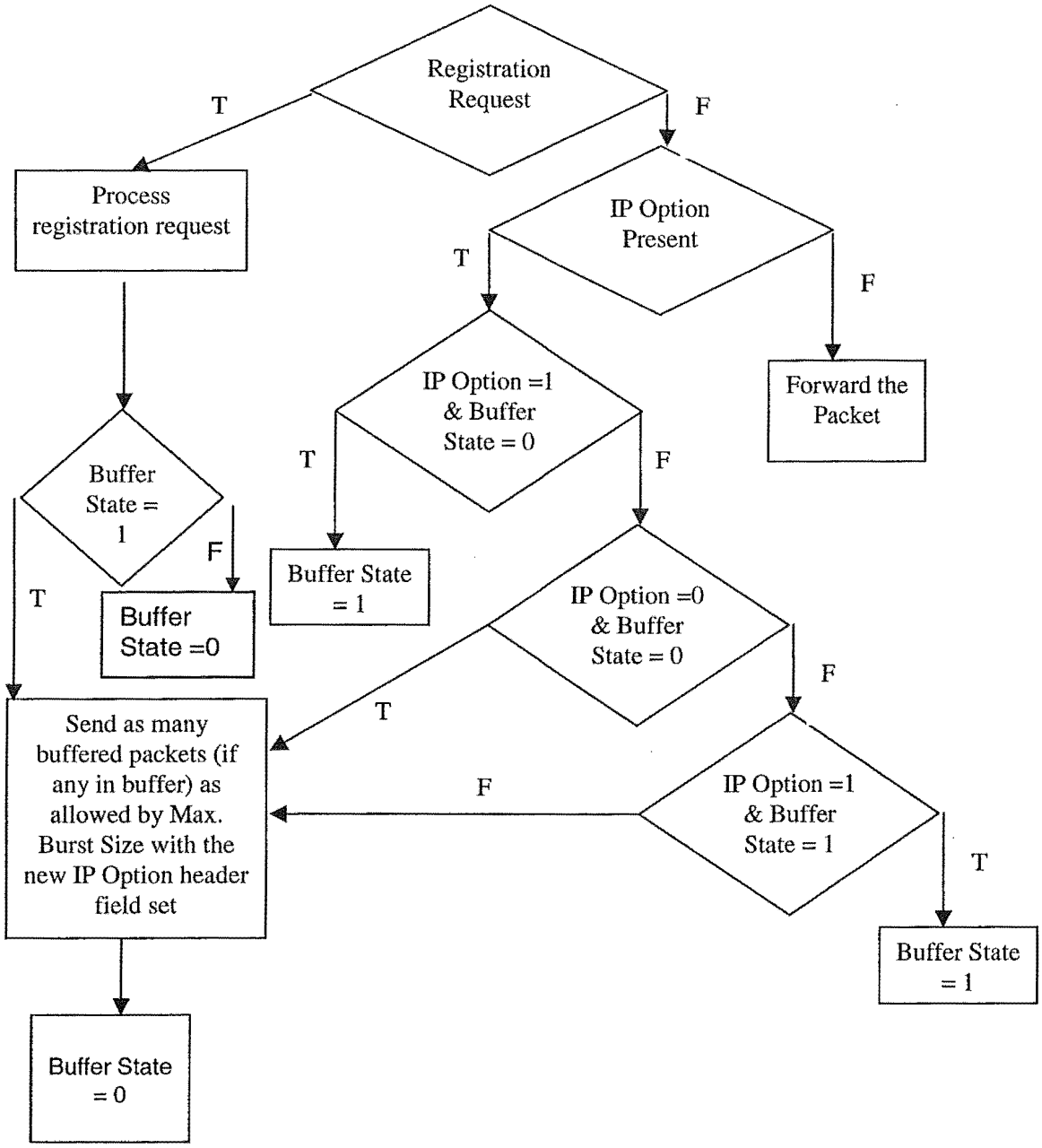

Figure 4.6.5.2-1: Summary of buffer agent actions relating to packets from the MH.

4.7. Summary

In this chapter, the mechanisms considered for the proactive disconnection detection and reconnection detection were described. Considerations to practical implementation issues were included. Issues that need to be further addressed for a future practical implementation were also highlighted. Modifications to the Freeze-TCP scheme were then proposed. These modifications focus on the issues of avoiding excessive idle times following the occurrence of losses due to disconnections.

Mechanisms to enhance the robustness of the end-to-end Freeze-TCP mechanisms were then discussed. These related to the addition of buffering at the wired network to minimise the performance dependence of the chosen disconnection prediction power threshold used. A method for using buffering while still allowing the use of certain types of encryption mechanisms was then proposed. This method used a new IP header option to avoid the need for snooping the TCP header. Finally, the issue of avoiding excessive bursts of data to the new BS following a handoff was addressed and a mechanism was described to deal with this issue.

The next two chapters present a simulation study of the Freeze-TCP mechanism and the mechanisms proposed in this chapter. The implementation of the buffering scheme is based on the buffer blocking concept described in this chapter.

Chapter 5 provides an overview of the simulator used and the methodologies adopted in the simulations. Chapter 6 then presents and discusses the results obtained.

Chapter 5. Experimental Methodology

This chapter describes the main aspects and limitations of the simulator used, and the modifications made to it. The specific simulated environment and the performance metrics used to measure and compare performances are also described.

5.1. Simulation Environment

5.1.1. Network Simulator Used

To simulate the environment described in Chapter 4, the ns-2 network simulator [Fall and Varadhan 2000] from the VINT Project at UC Berkeley, LBL USC/ISI and Xerox PARC was used. The version used was ns-2.1b7. This simulator is used extensively in the research community for the simulation of TCP performance. It contains implementations of many TCP variants, including an implementation of SACK based on TCP-Reno, called Sack1. In the simulator, the sequence numbers of TCP data sent and of ACKs relate to packet number instead of byte number.

This simulator version has a Mobile IP implementation as well as support for wireless simulation. This includes an implementation of the IEEE 802.11 distributed co-coordinated function (DCF) MAC protocol with RTS/CTS/DATA/ACK for unicast packets and only DATA packets for all broadcast packets. The simulator approximates the Lucent WaveLAN Direct-Sequence Spread-Spectrum (DSSS) radio interface.

The ns-2 wireless model includes the Free-Space, Two-Ray Ground and Shadowing propagation models [Rappaport 1996]. Since the base distribution only contains ad-hoc routing protocols, the NOAH (non-ad-hoc) routing model [Widmer 2001] was added to

our simulator installation. This routing model only allows communication to/from wireless nodes through a base station. This accords with our simulation environment described in Section 4.

The routing protocol was observed to significantly impact on the performance of TCP. We experimented with the Destination-Sequenced Destination-Vector (DSDV) routing protocol available in ns-2 in the early stages of the study and saw that it caused much larger disconnection periods than the NOAH protocol. Therefore, it was important to use the NOAH routing protocol as we seek to study cellular based networks rather than ad-hoc networks. In the future the use of the proactive window freezing mechanism could also be studied in the context of ad-hoc networks and it is expected to provide even better performance improvements over unmodified TCP due to the avoidance of serial timeouts during the route stabilisation period of the ad-hoc routing protocol.

5.1.2. Limitations of the Simulator

The main limitations of the ns-2 simulator with regard to the study were:

- No dynamic advertised window: In the ns-2 simulator, an advertised window is set at the start of the simulation and kept fixed throughout the simulation.
- No zero-window probes: Since the advertised window cannot be dynamically modified, there is no support for sending zero-window probes in case the advertised window reaches zero.
- When the advertised window is zero, the ns-2 simulator does not freeze its retransmit timer.

5.1.3. Modifications Made to the ns-2 Simulator

Besides the inclusion of the NOAH routing agent into the ns-2 simulator used, several modifications were also performed. The main additions were the capability to set the

advertised window to zero, based on the received signal power strength, and the ability to freeze the sender retransmit timer if a zero window advertisement is received by the sender. Zero window probes with exponential backoffs were also added.

The ability to send three copies of the ACK for the last in-order received packet upon handoff completion [Cáceres and Iftode 1994], [Goff et al 2000] was also added. These changes to the MH are explained in Section 5.1.4.

The buffer agent used in the Gateway FA buffer blocking mechanism described in Chapter 4 was implemented by modifying the Snoop agent and Priority Queue modules in the ns-2.1b7 source code.

5.1.4. Simulation Implementation

A disconnection prediction was implemented in the simulator using the fixed threshold method described previously where the received signal level of each TCP packet was monitored and a packet correctly received was compared to a fixed threshold and marked if the signal was below the warning power threshold chosen. The reception of a marked packet allowed the TCP code to send a ZWA for this segment and update the time the last ZWA was sent. The threshold value can be changed by the user in different simulations to test how sensitive the scheme is to the threshold used. Figure 5.1.4-1 summarises the MH actions in the simulation implementation when a TCP packet is received from the wireless link.

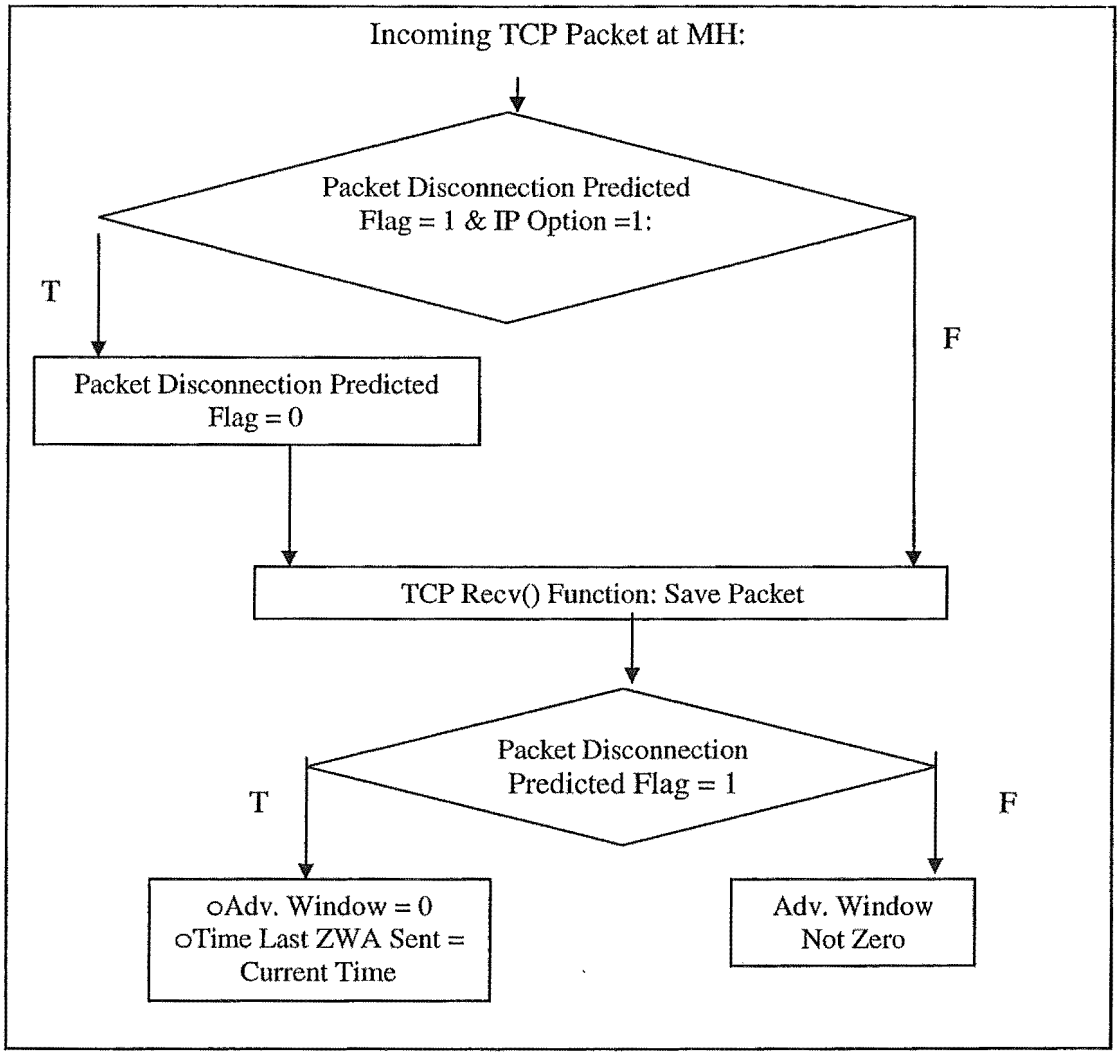

Figure 5.1.4-1: Summary of MH actions on receiving a TCP segment in simulator implementation.

In the ns-2 simulator used, a registration request is sent for every beacon received and the MH must periodically refresh its binding with the current BS. Hence a reconnection can be due to the reception of a registration reply following a handoff or following a disconnection due to the loss of registration caused by the loss of beacons.

To detect a reconnection in the simulator, the Mobile IP agent in the MH was modified so that it keeps a state (ChangedBS) that is set when the MH receives a beacon from the new BS or if a beacon from a BS at which the MH is not registered is received. If a

registration reply is received when the state is set, this indicates that a handoff has been completed. The TCP layer is then called to update the timestamp echo value according to Section 4.5 and send three copies of the ACK for the last packet received. The ChangedBS state is then reset.

The buffer agent implements the actions summarised in figures 4.6.3-3 and 4.6.5.2-1.

5.1.5. Simulation Assumptions and Simplifications

The Mobile IP implementation of ns-2 was used to simulate the handoff between two BSs in the same domain using a hierarchical cellular structure as shown in Figure 4.1-1. Since the Home Agent (HA) of Mobile IP is positioned at the central node between the two foreign agents, it corresponds to the Gateway FA of micro-mobility schemes such as Regional Registrations.

In order to isolate errors during handoffs, packets can only be lost due to BS buffer overflow or to the mobile not being within range. Therefore, the Free-Space propagation model was used. Hence the received power decays with the square of the distance between the transmitter and receiver [Rappaport 1996]. It is assumed that the cell range has a radius of 40m. Therefore, as long as the mobile is within this range all packets can be received without errors and when the mobile exceeds this distance from the BS, all packets are marked as in error and not received by the TCP layer.

In this thesis, we are not concerned with the methods used to measure the signal strength; we just assume that the lower layers can provide signal measurements and the information about the link status at the TCP layer represents the current link condition. Therefore, the method used to inform the TCP layer of the link condition in the simulator was to set/reset a new bit we added to the existing packet structure at the network interface. The TCP layer reads this flag on each packet received at the TCP layer to emulate the link state information received from the lower layers.

A few threshold power levels are chosen from measurements in the simulated network configuration to provide the effects being studied. The details of how this threshold value is chosen in a real implementation are not considered in this thesis since we concentrate on the effects in the TCP performance. We, however, look at the effect of changes in this threshold value on the obtained performance to see how much the scheme under study depends on a proper choice of threshold.

An ICMP reconnection message is also not implemented in the simulation model. Instead, the Mobile IP layer calls the TCP layer directly upon reconnection detection (Section 5.1.4).

5.1.6. Simulation Parameters

The default values of the ns-2.1b7 simulator are used unless otherwise stated. A TCP packet size (including TCP/IP headers) of 1000 bytes is used while the acknowledgement packet size is 40 bytes.

In order to allow the occurrence of many disconnection events, simulations are run for 100 seconds and data transfer starts after 10 seconds of simulation to avoid simulation start-up effects. This would be more important when using ad-hoc routing protocols, where a period of time is required for route stabilisation at simulation start, but it is also used in our non-ad-hoc case.

Although we consider a pedestrian mobility scenario (the mobile moves with a speed of 1.5m/s), handoffs are performed each 5 seconds and simulations are run for 100 seconds. This small cell latency provides a number of disconnection events in a simulation run. The use of five seconds cell latency was motivated by the desire to study the effect of disconnections in general in a controlled manner. A disconnection can happen not only due to a handoff but also due to temporary fading.

The link speed and delay parameters are those shown in Figure 4.1-1 unless otherwise specified and the number of background sources is varied to test the effect of different congestion levels. The default BS buffer size has capacity for 20 packets in our simulations and when a different value is used in one or more buffer in the simulated network attention is drawn to this fact.

Since some tests were conducted in Chapter 6 using a small buffer size of three packets, we used a burst limit of two packets in our simulations to test the ability of the mechanism in reducing the congestion levels caused by bursts of data. A limited set of tests was initialled conducted in the presence of background traffic and two TCP connections using burst limits ranging from one to three packets. Although using a burst limit of three packets from the Gateway FA buffer agent allowed performance gains over the case where the entire buffer contents were sent to the new BS, the performance was not as high as when the burst was limited to two packets. Hence we used a burst limit of two packets throughout the burst control simulations to see how it affected the TCP performance in different scenarios, both where the new BS was congested and where the new BS was not congested.

As highlighted in [Allman and Falk 1999], in order to provide more significant results in terms of real environments, tests should include not only a single TCP connection but also the presence of background traffic and competing TCP connection. Hence we include background traffic in most tests presented in Chapter 6. In some cases no background was used to provide for more controlled conditions.

The background traffic used is not an attempt to model real background traffic that would be encountered in practice as the traffic in the Internet is highly variable and difficult to model. Instead, a background traffic that simply provides some burstiness is used. The level of background traffic is chosen to avoid congestion losses when we would like to avoid such losses in order to be able to isolate performance problems caused by other factors, such as handoffs. However, we also evaluate the effect of congestion on the

window freezing scheme, particularly in order to see the effect of using the old window in a congested cell. This will be discussed in detail in Chapter 6.

Either two or four Pareto sources are used to provide different levels of congestion in the network. When background sources are used, each MH receives background traffic from either one or two sources, depending on whether two or four background sources are used, respectively. The number of background sources used is specified when we present the scenario used for each test in Chapter 6

Table 5.1.6-1 summarises the main parameters used in the simulations in Chapter 6. As previously mentioned, these are not intended to follow a particular real scenario, but rather to highlight observations about strengths and weaknesses of the different schemes.

Main Simulation Parameters	
Parameter	Value
TCP Packet Size	1000 Bytes
ACK Packet Size	40 Bytes
Background Packet Size	200 Bytes
Pareto source on time	0.5 s
Pareto source off time	0.5 s
Pareto Shape	1.2
MH Speed	1.5 m/s
Time between handoff events	5 s
Cell Range	40 m
Propagation Model	Free Space

Table 5.1.6-1: Summary of simulation parameters.

5.1.7. Statistical Analysis of the Simulation Results

The use of bursty background traffic, the fact that the beacons are sent with a uniform distribution between zero and one and the use of competing TCPs in some of the tests presented in Chapter 6 required the use of statistical analysis for proper interpretation of the results as well as comparison among the schemes performances.

We used a sequential analysis program[1] to run multiple simulations and the throughput of the simulation runs was used to decide when to stop the simulations. This program initially executes five simulation runs and then the mean throughput is calculated. After this the relative error is calculated and if found to be above 5% a new simulation run is performed. The new mean is calculated and the process is repeated until the relative error of the mean throughput is less than or equal to 5%.

For most scenarios using a single TCP connection approximately five simulation runs were enough to achieve the desired error target. However, when two competing TCP connections were used, considerably more runs were needed in some cases to achieve the target statistical error.

Since the TCP-SACK had a higher throughput variation than the window freezing schemes tested, in case of two competing TCP connections the sequential analysis program used the throughput of the TCP-SACK connection to decide when to terminate the simulation process.

When comparing two results that had a small percentage difference, statistical tests were also performed to decide whether or not such difference was statistically significant, given the variability observed. Details about the statistical test used can be found in Appendix B.

5.2. Choice of Baseline TCP

The baseline TCP chosen served as the basis of comparison with the modifications made in this work. We do not consider the use of Delayed Acknowledgements in this study.

[1] Developed at the University of Canterbury by Sven Ostring and Associate Professor Krzystof Pawlikowski.

The timestamp option [Jacobson et al 1992] is turned on in our tests unless otherwise specified.

The baseline used is TCP-SACK following [Mathis et al 1996] on top of the TCP-Reno implementation. This is the TCP-Sack1 implementation available in the ns-2 simulator. This is a conservative SACK implementation in that it makes minimal changes to the TCP-Reno congestion control algorithm. The TCP-Sack1 implementation and its performance under multiple packet losses are described in [Fall and Floyd 1996].

The use of SACK is growing fast [Floyd 2001] and has been shown to be beneficial when multiple packets are lost in a window of data [Fall and Floyd 1996], such as in wireless environments [Balakrishnan et al 1997], [Kuhlberg 2000]. Hence in our environment SACK was used whenever possible.

It should be noted that in the TCP-Sack1 implementation of TCP-SACK of ns-2.1b7 and newer versions, a fast retransmission is not allowed if the SACK blocks in the DUPACKs do not acknowledge new data [Blanton 2000]. This change was proposed to avoid situations where packet reordering and/or ACK duplication in the network leads to unnecessary fast retransmissions. Since we send three copies of the ACK (TR-ACK) for the last data segment received following a reconnection to unfreeze the sender as in [Goff et al 2000], this means that the TR-ACK will not trigger a fast retransmission at the sender. Such behaviour may be a disadvantage in cases where the mobile has entered a more congested cell and reducing the congestion window would be advisable.

5.3. Performance Metrics

In measuring the performance of TCP, we consider the segment size (TCP + IP headers lengths + Data length) when measuring the number of bytes received by the receiver since the simulator version used does not make distinction between TCP/IP headers and TCP payload. Although the main performance metric of interest was the receiver

throughput, other performance metrics were also recorded to help in the analysis of the results. Trace files of the sender, receiver and the evolution of the congestion window were also analysed on many occasions to help in understanding the behaviour of protocols in the scenarios simulated.

5.3.1. Average Receiver Throughput

This metric is defined as the total number of bytes correctly received at receiver divided by the total time the source was active. For simplicity, this metric is referred simply as throughput for the remaining of the thesis.

5.3.2. Goodput

This metric is defined as the quotient of the number of bytes correctly received by the receiver and the number of bytes sent by the sender.

5.3.3. Number of Timeouts

Since one of the most important objectives in a wireless scenario is to avoid unnecessary timeouts, the number of timeouts that occurred over the connection in question was also recorded. This makes it possible to see to what extent it was possible to avoid timeouts in the scenarios tested and the relative occurrence of timeouts in the protocols tested.

The use of deterministic handoff times allowed this metric to be used to observe how often timer backoff occurred. Since the number of handoffs during a simulation is known, it was possible to see, for example, that timer backoff occurred several times in some cases where the number of timeouts observed was higher than the number of handoffs and enough network capacity existed to avoid congestion related losses.

5.3.4. Number of Retransmitted Bytes

The comparison of the number of retransmitted bytes and the number of timeouts allowed us to observe situations where no timeouts occurred but losses still occurred and recovery was entirely through fast retransmits/fast recovery.

5.4. Summary

This chapter presented an overview of the simulator used for the study of the performance of mechanisms discussed in the previous chapter. Its limitations relating to our study were highlighted and the modifications we implemented to overcome these shortcomings were presented. The choice of the baseline TCP used for comparison was also explained and its general characteristics reported.

The methodology utilised for analysing the results, the major performance metrics of interest and the main simulation parameters used were included. The next chapter presents the simulation results obtained for several scenarios used to study different aspects of the window freezing schemes considered in Chapter 4. The extent of the problems observed will be largely dependent on the specific scenario and hence generalisations are hard to make. However, the results shown can provide an idea of the expected behaviour of the schemes in real situations, as well as their strengths and weaknesses.

Chapter 6. Performance Evaluation and Analysis

This chapter aims at evaluating strengths and weaknesses of the base Freeze-TCP (Freeze-TCP) and of the proposed modifications discussed in Chapter 4. Comparisons are made among the performance of TCP-SACK and Freeze-TCP. Freeze-TCP is also compared with Freeze-TCP enhanced with the proposed proactive buffer blocking mechanism at the gateway (Freeze + Gateway Blocking) and timestamp echo updates (Freeze + Gateway Blocking + TS Echo Updates). Performance evaluations relating to the benefits or limitations brought by the use of burst limit at the Gateway FA are included.

Specific questions we seek to answer in the environment studied are:

- How will different thresholds affect the average throughput and how will different delays affect the performance for a given threshold?
- How will new acknowledgments received by the sender after a disconnection impact on the recovery process when a timeout is required to restart the sender?
- How will the window freezing mechanism impact on a competing TCP?
- Restarting the sender at full speed after a handoff could be inappropriate since the sender may have an inappropriate window for the load in the new cell. Is this observed in the environment studied and how do the schemes deal with this situation?

First of all, the performances of the window freezing schemes and TCP-SACK are compared under different link delays. More detailed analysis is then conducted including background traffic and using a fixed link delay. The effect of the threshold choice on the window freezing schemes is then studied. Following this, a more detailed simulation study on the problem of long idle periods, due to new ACKs

received after a reconnection, is performed. Simulations are also performed to see the interaction between the window freezing mechanism and unmodified TCP. Finally, simulations are performed in a scenario where cells have different load levels to evaluate the effect of unfreezing the window with the same window size as that used prior to the handoff. This also serves to study the interaction with unmodified TCP.

As explained in Chapter 5, the simulations are not intended to reproduce any specific real scenario but rather to provide a better understanding of how the schemes studied work under controlled conditions to facilitate the analysis and allow meaningful conclusions to be drawn about strengths and weaknesses of the schemes considered.

6.1. Performance Dependence on RTT

The aim of this section is to observe how robust the window freezing without buffering at the Gateway FA is, and how the addition of Gateway FA buffering can aid in making the Freeze-TCP more robust to different delays given a certain fixed threshold. In this section the smallest threshold tested is used for various link delays and the throughput of different schemes is compared. This corresponds to when the MH predicts a disconnection if the received power of a successfully received packet is below 0.5% the minimum receivable power. To isolate the effect of the warning power threshold used and provide a controlled RTT, no background traffic is present in these tests and the sender is at W(4) in Figure 4.1-1. A single MH is present in this scenario.

Figure 6.1-1 shows that in this scenario, the window freezing schemes with or without buffering performed similarly when the wired link delay between W(0) and W(4) is small. As the link delay starts to grow, the Bandwidth x Delay product [Stevens 1994] grows and the throughput starts to become limited by the receiver advertised window. The recovery from multiple losses during the disconnection times is also delayed in the case of the pure end-to-end schemes due to the increased link delay. Therefore the throughput decreases as the delay increases.

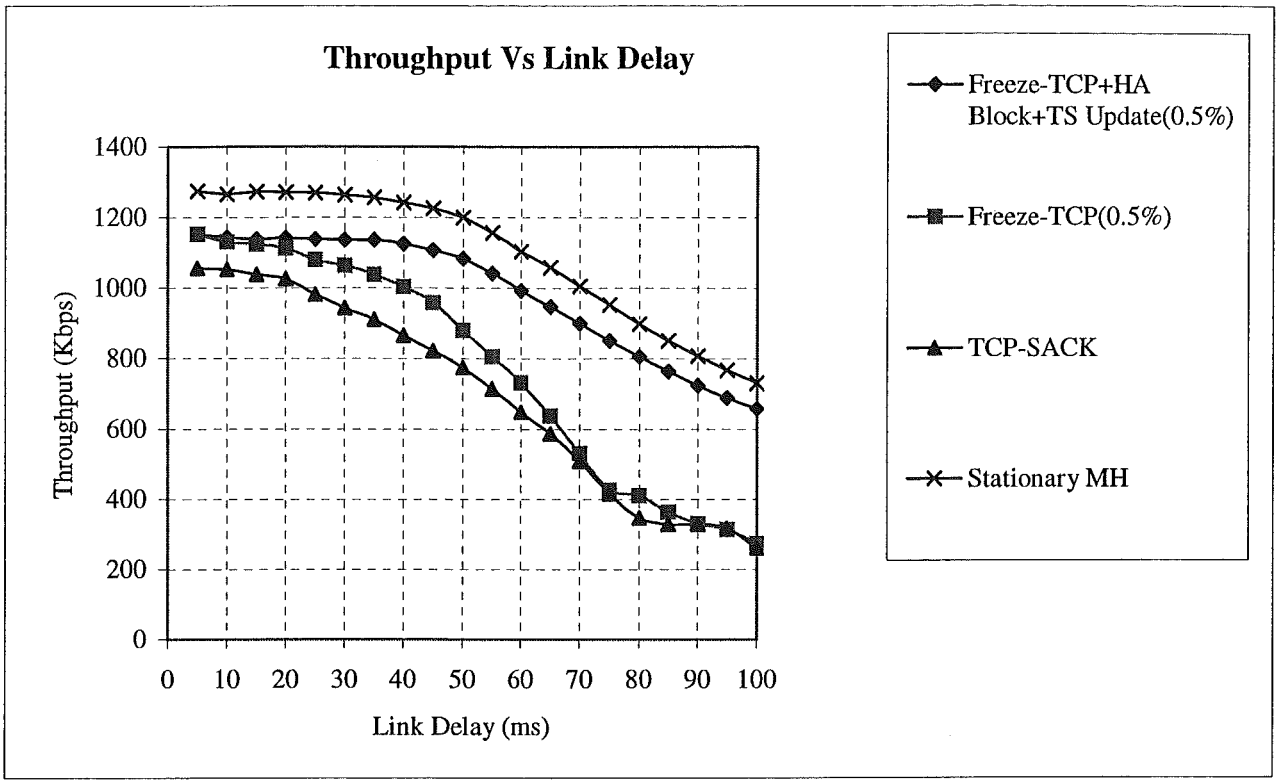

Figure 6.1-1: Throughput vs Link Delay for TCP-SACK, Freeze-TCP and Freeze-TCP with Gateway FA Buffering.

Furthermore, as the link delay increases, there is less time for the ZWA to reach the sender before the disconnection occurs and the performance of the Freeze-TCP starts to approach that of the unmodified TCP-SACK. On the other hand, when buffering in response to a disconnection prediction is used at the Gateway FA, the robustness of the prediction mechanism is increased and the need for end-to-end retransmission is reduced. Therefore Freeze-TCP with the addition of buffering at the Gateway FA had a significantly slower throughput reduction as the link delay was increased than the pure end-to-end schemes.

When the delay is small there was no advantage in performing buffering at the Gateway FA since the difference between the sender to Gateway FA and Gateway FA to MH delays are not large. Freezing the window, however, still provides improvements over TCP-SACK since the sending rate can be restarted at full rate if the disconnection prediction is successful.

Similar trends were observed for the case where the Gateway FA kept a copy of the packets sent after a disconnection prediction and for the case where the Gateway FA blocked the buffer after a disconnection prediction is received. No difference in throughput versus link delay performance was observed when adding our timestamp echo update scheme and a burst limit of two packets to the scheme with only buffer blocking in response to a disconnection prediction from the MH in this scenario.

As opposed to the end-to-end Freeze-TCP scheme, the use of buffering in this scenario avoided losses during the handoffs for all link delays considered. Since no handoff related timeouts are observed, the timestamp update is unnecessary when buffering is used in this scenario. Likewise, the lack of competing traffic and the large buffer sizes used made unnecessary limiting the burst sent from the Gateway FA following the handoff completion. However, doing so did not cause noticeable performance degradation either.

6.2. Effect of Threshold Choice on Throughput

Since a fixed warning threshold power is used, it is important to see how its choice influences the performance obtained. In this section, we used the scenario of Figure 4.1-1 where four background sources with the configuration described in Chapter 5 are present. A single TCP connection is used.

6.2.1. Ideal Link with Background Traffic

In this case the MH is stopped within one cell and only one TCP connection is used. This was used to verify that the background traffic used would not introduce congestion related losses to the single TCP connection case and to investigate what the maximum throughput achievable in our configuration could be.

The average throughput observed in these conditions was 701.4 Kbps and no losses were observed in the TCP connection in this scenario. This throughput is considerably lower than the slowest link throughput and is due to the advertised window size being

smaller than the pipe capacity. This causes the maximum throughput to be limited by CWND/RTT rather than the slowest link bandwidth.

6.2.2. Effect of Warning Power Threshold on Throughput

Although the choice of this threshold is dependent on the particular receiver characteristics or MH speed, and we do not seek to find a general appropriate threshold value, the objective of this test was to investigate the extent to which the warning threshold power influences the performance obtained in our environment. This test also served as a basis for choosing the threshold used in other simulations presented in this thesis according to the test objectives. During these tests the problem of inflated RTO (Section 4.4) and consequent performance degradation was also uncovered.

Throughput

Figure 6.2.2-1 shows a comparison of Freeze-TCP, TCP-SACK as well as Freeze-TCP with Gateway FA buffer and timestamp update for different warning threshold power levels. Also included in the comparison is the maximum throughput achieved in this scenario. This corresponds to the case where the mobile is stopped throughout the connection duration and no losses are introduced. The reminder of this section discusses the results in Figure 6.2.2-1.

This figure indicates that Freeze-TCP can provide substantial performance gains in our network topology if the threshold is sufficient to give the sender time to allow ceasing transmission and therefore allowing all in-flight data to reach the MH before the disconnection. This is consistent with the findings in [Onoe et al 2001].

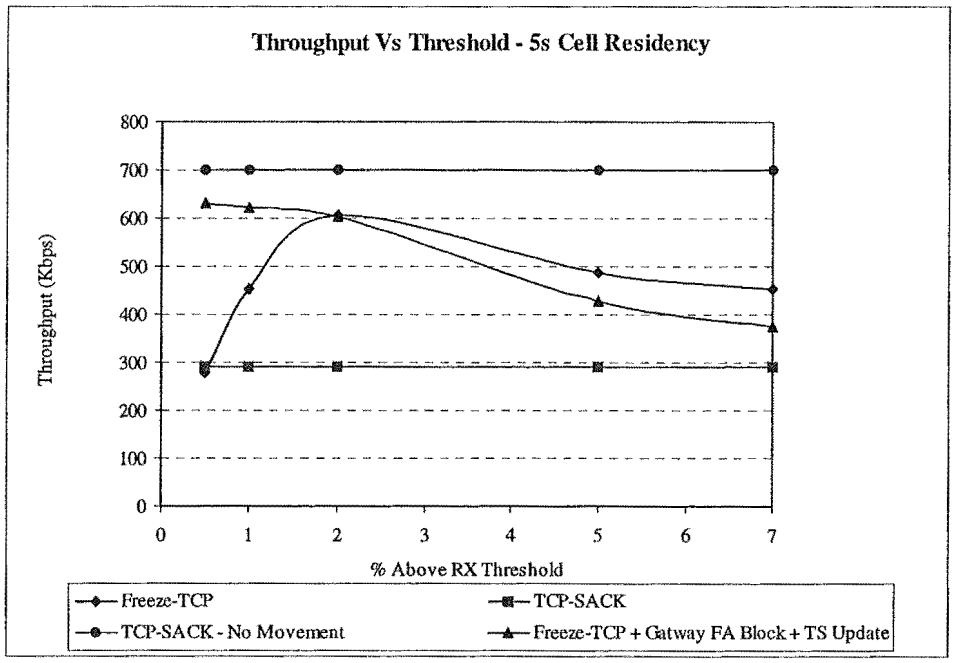

Figure 6.2.2-1: Effect of different warning power threshold levels in the window freezing mechanisms performance.

Obviously, the extent of the performance gains from freezing the congestion window and the performance penalty that may be incurred by late freezing are dependent on the level of losses in the wireless link; if the window is kept small due to frequent errors, the benefit of avoiding window drops are reduced.

From Figure 6.2.2-1 it can be seen that a warning power threshold 2% above the minimum receive threshold provides the best throughput in our test environment for Freeze-TCP.

Losses and Retransmissions

Observation of Figures 6.2.2-1 and 6.2.2-2 shows that using a warning threshold power above 2% of the minimum receivable power led to worse performance than if the former warning threshold was used. This indicates that using such higher warning thresholds led the sender to be frozen prematurely. On the other hand, warning threshold powers smaller than 2% above the minimum receive threshold power could not avoid losses to Freeze-TCP.

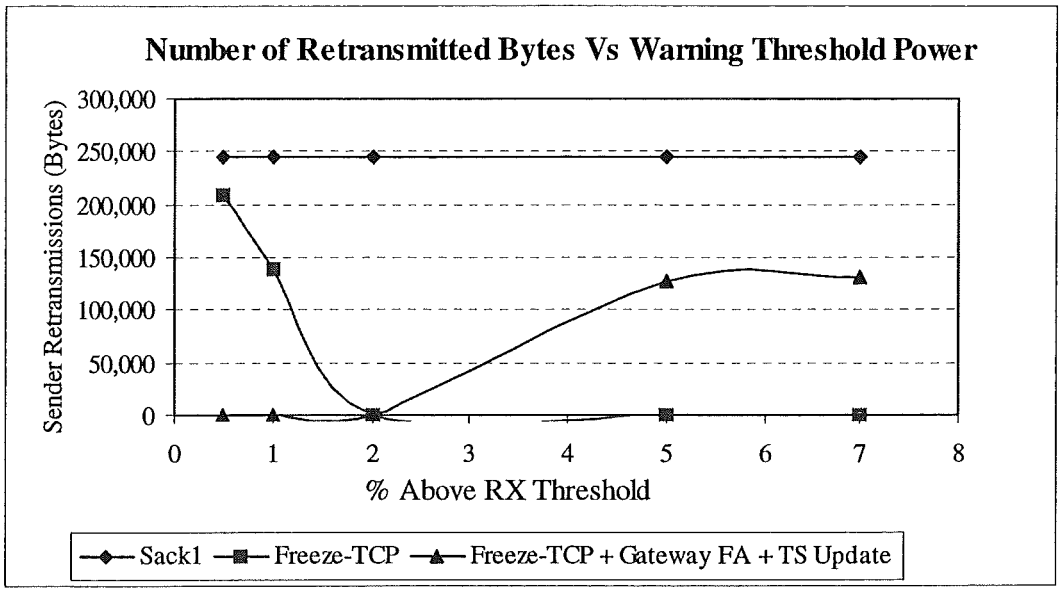

Figure 6.2.2-2: Number of retransmitted bytes Vs warning threshold power level.

Although Freeze-TCP was unable to improve the receiver throughput in relation to the TCP-SACK case when the lowest warning threshold power was used, the former scheme had less than half the number of timeouts seen in the TCP-SACK tests. This indicates that in this case Freeze-TCP was capable of freezing the window and the retransmission timer but still suffered losses. This situation will be examined in more details in Section 6.3.

Losses Through Early Buffer Unblocking

One of the main objectives of buffering at the network in preparation for a handoff is to make the window-freezing scheme more robust to the time required to send the disconnection warning to the sender and so avoid loss of in-flight segments. Also, as stated by [Onoe et al 2001], buffering at the network should allow for smaller warning thresholds to be used. This avoids freezing the window prematurely.

Figures 6.2.2-1 and 6.2.2-2 show that, as expected, the addition of the Gateway FA buffer mechanism leads to better performance than Freeze-TCP in cases where Freeze-TCP would not have enough time to avoid loss of in-flight data. However, the

figures also show that the buffer blocking implementation achieved a worse throughput than Freeze-TCP and required retransmissions at the sender when high thresholds were used. The reason for this performance deterioration is the sender unfreezing prior to the handoff and the consequent loss of data and congestion action by the sender. An example of this behaviour is shown in Figure 6.2.2-3 where the sender unfreezes the congestion window soon before the disconnections at times 30 seconds and 40 seconds leading to losses and the consequent congestion window reduction due to timeouts.

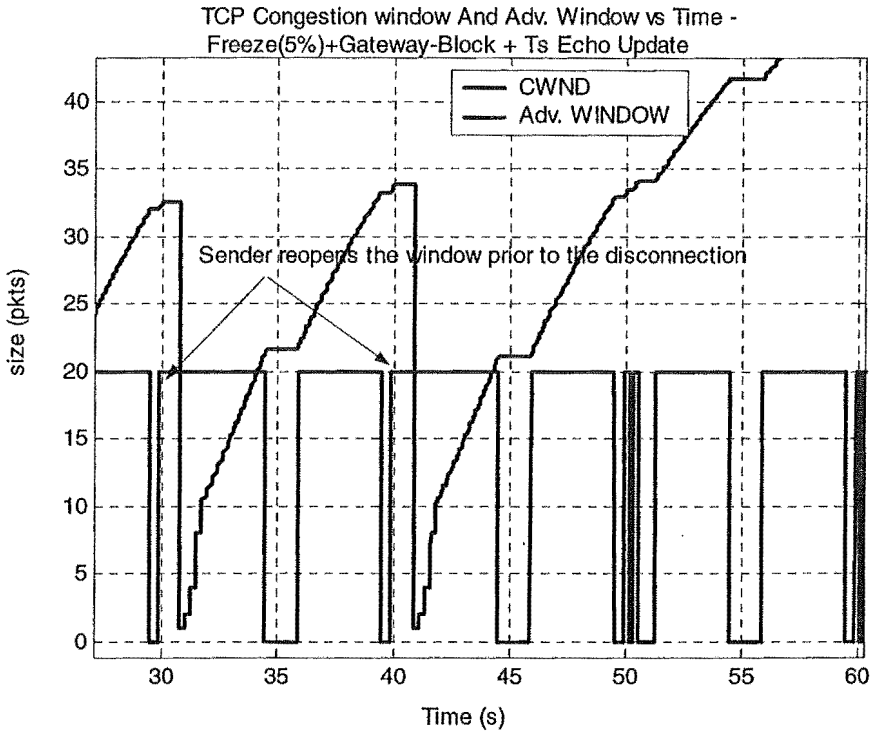

Figure 6.2.2-3: Congestion Window unfrozen prior to the disconnection leading to losses.

The sender can be taken out of persist mode if a registration request is received by the buffer agent, which unblocks it. Buffered packets do not generate a ZWA in order to avoid unnecessarily freezing the sender again while the MH is moving away from the BS. Hence the ACKs for the buffered packets will take the sender out of persist mode. However, if this is done close to the time the handoff occur, packets sent after the sender is unblocked are lost. Therefore, the sender may be frozen prematurely, wasting the opportunity to send data, and still require timeouts for recovery of losses.

Note that since the sender was unfrozen prematurely timer backoff can still occur. Hence this scheme, if used with excessively large warning periods, can lead to performance degradation and in certain cases may perform even worse than the unmodified TCP due to the unnecessary freezing of the window and congestion action.

Therefore it is important for the MH to know whether the Gateway supports buffering before the threshold is chosen. This can be done at power up and the MH will only use a low power threshold if Gateway buffering is supported.

In the case of [Onoe et al 2001] scheme's, this problem would not occur because a more conservative approach is used. In that scheme, the buffered data is only sent if the BS sees that the signal from the MH is above the threshold and a non-ZWA is sent by the MH only after it detects a signal above the warning threshold. Hence the sender is not prematurely unfrozen, but it may also need to wait for a longer time until a beacon with signal above the threshold is received, specially if the beacon period is large and a beacon with signal below the power low threshold is received.

Since our implementation is at a node carrying more load than a single BS, it is reasonable to try to send buffered data as soon as there is an indication that the link to the MH is active to minimise the buffering requirements.

The reason for the sender being repeatedly frozen and unfrozen at around time 50 seconds in the Figure was also due to the high power threshold used in this case. The sender is prematurely frozen and unfrozen as described above. Since buffered packets do not generate a ZWA as previously described, these also unfreeze the sender. However, the packets sent when the sender is unfrozen are received at the MH and generate a ZWA due to the high threshold used. This process is repeated several times while the MH is within the low power warning region.

6.3. Recovery Performance after a Handoff with Losses

In this section the extent to which the RTO inflation problem following a reconnection occurs in our Freeze-TCP simulations is investigated in the context of

the scenario used in the last section. First the problem is analysed through the study of sender traces following a reconnection. Then the extent to which the RTO inflation affects our main performance metrics, namely the average receiver throughput, is investigated in different scenarios. The configuration used is that of Figure 4.1-1 with four background sources and a single TCP connection used.

To provide a better comparison among the schemes studied here, a statistical error of 3% was used in this case instead of the 5% statistical error used to control the sequential analysis program in the other cases.

6.3.1. Sender behaviours following a reconnection

In order to show the performance penalty after the sender is unfrozen but data segments and the last ACK sent by the MH were lost during the disconnection period, traces of TCP-SACK, Freeze-TCP and of Freeze-TCP with the proposed timestamp echo update modification are presented. These traces show the inactive time at the sender due to the mobile moving out of the coverage area of FA1 in Figure 4.1-1 and into the coverage area of FA2 at time 55 seconds.

TCP-SACK

First we present the recovery of the unmodified TCP-SACK following a handoff 55 seconds into the simulation. Figure 6.3.1-1 shows that for TCP SACK two timeouts occurred due to the handoff and the fact that the sender timer was not frozen. The exponential backoff causes unnecessary inactive times at the sender. From this figure it can be seen that one second elapsed between the first and the second timeout due to the exponential backoff of the timer following the first timeout.

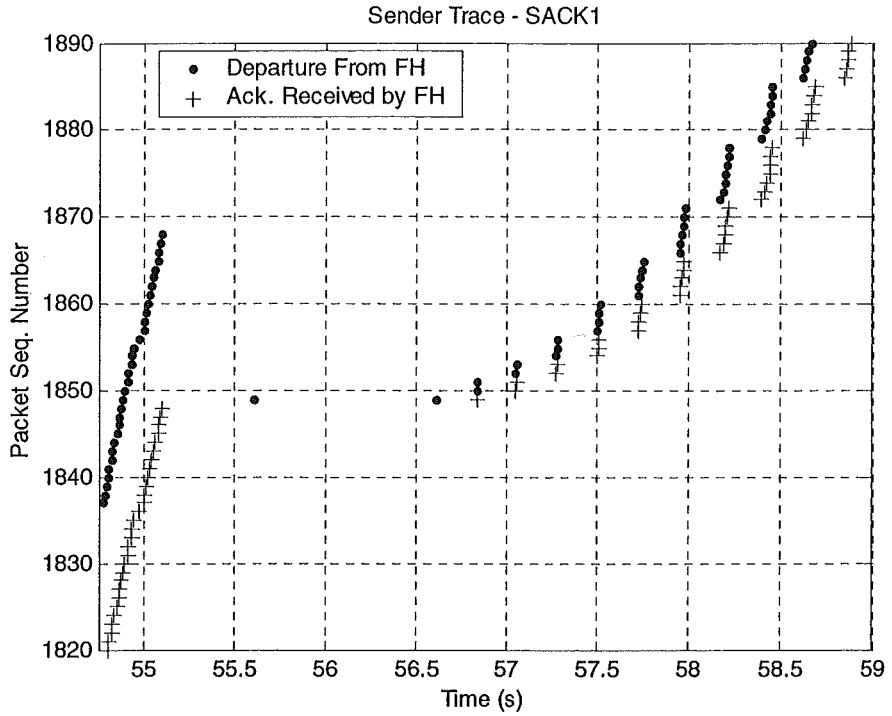

Figure 6.3.1-1: Sender trace zoom showing recovery time for TCP-SACK after a handoff.

Freeze-TCP

The situation where packets are lost due to late sender freezing and the time taken to restart the sender following the handoff is illustrated in Figure 6.3.1-2. In this case a warning power threshold 0.5% above the minimum receive power threshold was used. This caused the ZWAs to be sent too late to avoid the loss of in-flight data and the last ACK is lost as it is sent to the old BS with which the MH has already lost contact.

Figure 6.3.1-2 shows that the TR-ACK sent by the receiver unfreezes the sender, which sends a burst of new packets. However, because of the loss of data due to the late freezing of the sender, DUPACKs are returned for these packets and a fast retransmission takes place. Since the first ACK in the TR-ACK acknowledges new data, the RTO is updated. However, since the ACK is for a packet sent before the disconnection, the RTO will be unnecessarily increased. This is the reason of the large delay for the timeout, which the sender needs to wait for, since there were

multiple losses in the window of data and hence it cannot recover through the fast retransmission.

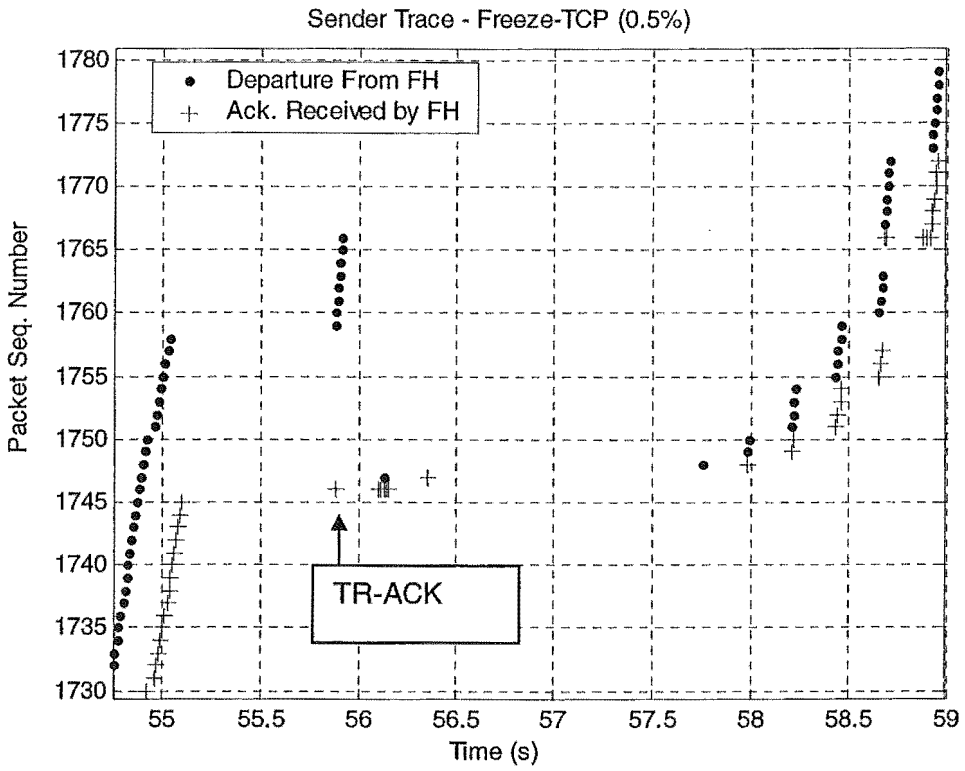

Figure 6.3.1-2: Zoom of sender trace showing problem caused by new ACK unfreezing the sender.

Note that approximately 1.6 seconds were required for the timeout to occur after the fast retransmit took place, even though there was no exponential timer backoff due to multiple timeouts. We would like to restart normal transmission as soon as possible after the fast retransmission since the link now is capable of receiving data. TCP-SACK (Figure 6.3.1-1) on the other hand had two timeouts and a RTO of 1.0 second after the timer backoff following the first timeout. This shows that Freeze-TCP had to wait for a significant unnecessary idle time for the timeout.

Freeze-TCP + TS Echo Update

Figure 6.3.1-3 illustrates a trace of the sender in the same configuration of Figure 6.3.1-2 but now including the timestamp update procedure described in Section 4.5.

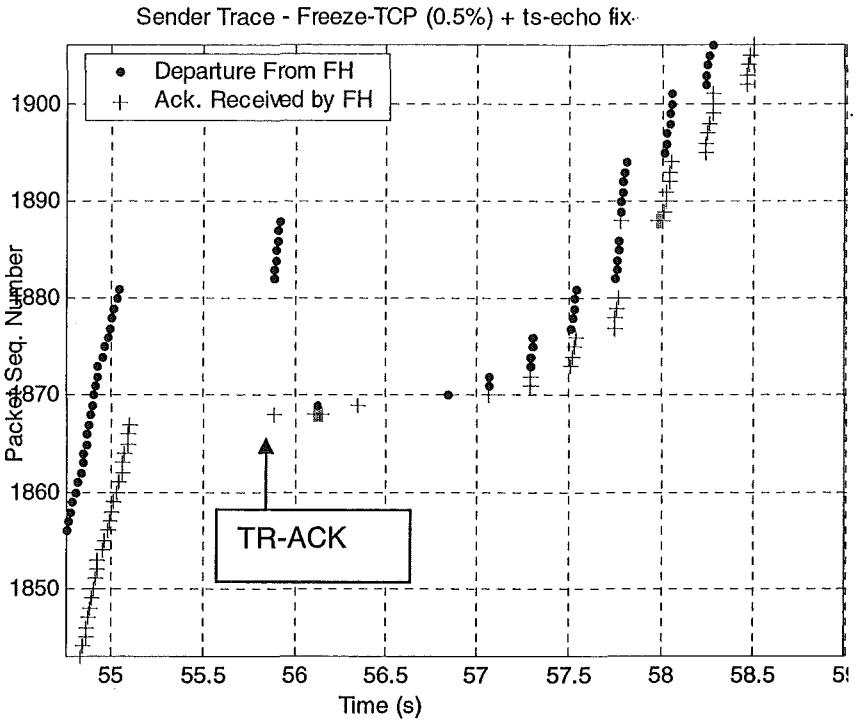

Figure 6.3.1-3: Sender trace zoom showing the recovery after sender is unfrozen by a new ACK using timestamp echo updates.

In the sender trace of Figure 6.3.1-3 the timeout that restarts successful communication occurs less than a second after the fast retransmission even though the first ACK in the TR-ACK received after the handoff also acknowledges new data.
Figure 6.3.1-2 and Figure 6.3.1-3 show that most new packets sent by the sender after reception of the reconnection warning ACKs (TR-ACK) and before the fast retransmission are still retransmitted, even though they have arrived at the receiver. This is due to the TCP go-back-N and slow start policy following a timeout.

6.3.2. Average Receiver Throughput Performance Simulations

Figure 6.3.2-1 shows a comparison of the mean throughput obtained for TCP-SACK, Freeze-TCP and the modified Freeze-TCP for a number of simulation runs in the scenario studied. Results are presented for adjacent cells and for one-second cell separation [Cáceres and Iftode 1994]. The 95% confidence interval of the results are summarised in Table 6.3.2-1.

Since there are multiple handoffs in each simulation run, the last ACK may or may not be lost depending on the time it is sent by the MH prior to the handoff, as in real disconnection situations. The small inter-handoff time is used to allow for many handoff events within the limited simulation time and no losses occur except due to the handoffs.

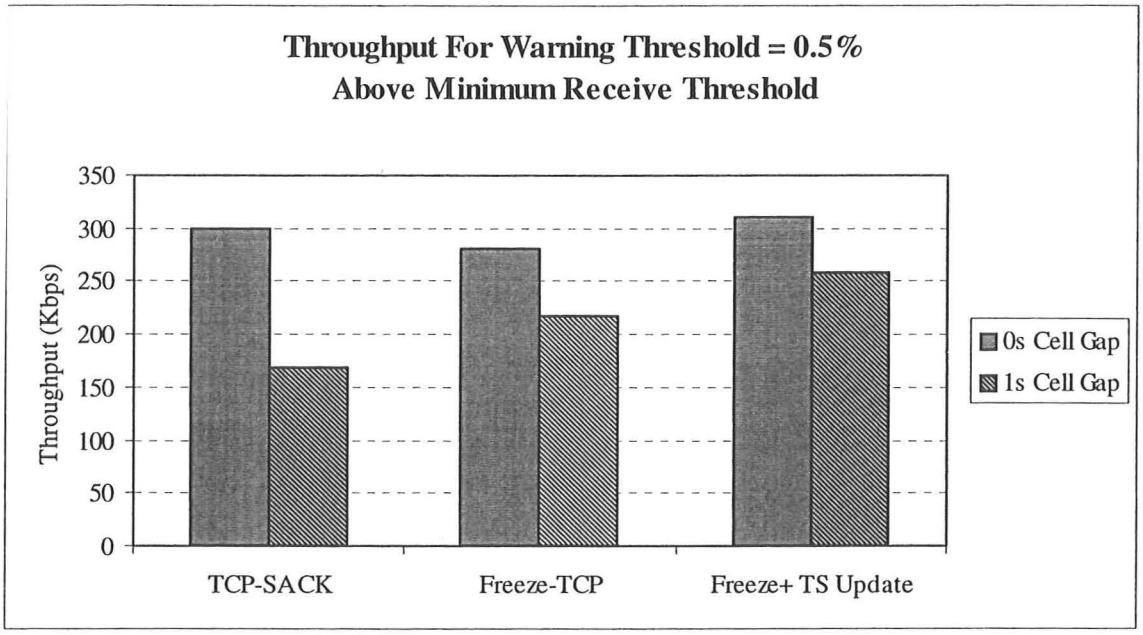

Figure 6.3.2-1: Throughput comparison of TCP-SACK and Freeze-TCP with and without timestamp update: handoff losses.

Table 6.3.2-1 shows that for the adjacent cells, the Freeze-TCP implementation performed on average 6.2% worse than the base TCP (TCP-SACK). The modified Freeze-TCP, on the other hand, performed on average 4.0% better than the TCP-SACK implementation and on average 10.9% better than the unmodified Freeze-TCP.

When the disconnection period is increased, the unmodified Freeze-TCP still perform better than TCP-SACK because of the higher penalty suffered by TCP-SACK due to timer backoffs during the longer handoffs. This can be explained by the fact that in the 1s cell gap, there were almost twice as many timeouts as in the adjacent cells case for TCP-SACK. On the other hand, the number of timeouts was practically unchanged

for the unmodified and modified Freeze-TCP tests in the adjacent and one second cell gap cases (on average nine timeouts during the simulation).

Scheme	Throughput (Kbps) - Adjacent Cells			Throughput (Kbps) - One Second Cell Gap		
	Min.	Mean	Max.	Min.	Mean	Max.
SACK	291.5	299.5	307.5	167.4	167.6	167.8
Freeze-TCP	273.3	280.8	288.2	211.4	216.9	222.0
Freeze+TS Echo Update	310.0	311.4	312.8	253.7	258.0	262.0

Table 6.3.2-1: 95% Confidence intervals of schemes performance in handoff losses scenarios for adjacent cells and one second gap between cells.

The performance improvement of the modified over unmodified Freeze-TCP is increased to 18.9% when a one second cell gap was used as can be seen in Table 6.3.2-1. Such improvement can be explained by the fact that in this scenario a larger disconnection time exist and hence the new ACKs received by the sender after a handoff will increase further the RTO. This will lead to worse performance when the sender needs to recover from the multiple losses due to the handoff with a timeout.

6.4. Competing TCPs and Implications of Data Bursts

A concern about using persist mode raised in [Vaidya 1999], was the potential problem due to the use of a congestion window that is larger than the appropriate value for the new cell after a handoff. In order to test this effect, we simulated a scenario with TCP connection (TCP-SACK) stopped at one cell while the other mobile moved periodically between cells.

Hence the window freezing schemes of interest were used in the mobile connection. Again, the best warning threshold studied was used so that the performance of the window freezing TCP can be maximised and the potential problem emphasised. Tests

were also conducted using TCP-SACK in the mobile connection for comparison purposes.

The buffer size at FA1 was 20 packets. When the mobile arrives at the cell that contains the competing TCP connection, its rate will be too large for the capacity available and hence lead to congestion at the new cell if its buffer is not able to avoid the congestion due to the window freezing connection.

This experiment was performed for buffer sizes of 20 and 3 packets at FA2, corresponding to large and small buffer sizes [Sarolahti 2000] respectively, so that different congestion levels will occur due to the sudden injection of packets at a high rate after the handoff. Two and four sources of background traffic were used to test different load levels in each buffer configuration.

The simulation results are first analysed in terms of the effect observed in each buffer configuration scenario. Comparison among different schemes is also discussed. Complete test results and the percentage improvement and penalty comparison among schemes can be found in Appendix A.

6.4.1. Equal Buffers Scenario

This section analyses the results of simulations in the scenario where both BSs had equally large buffer sizes of 20 packets. Hence enough buffer space was available on both BSs to limit the effect of congestion-related losses due to large congestion windows following a handoff.

Figure 6.4.1-1 and Figure 6.4.2-1 show the simulation results for the case where the BSs have the same buffer sizes.

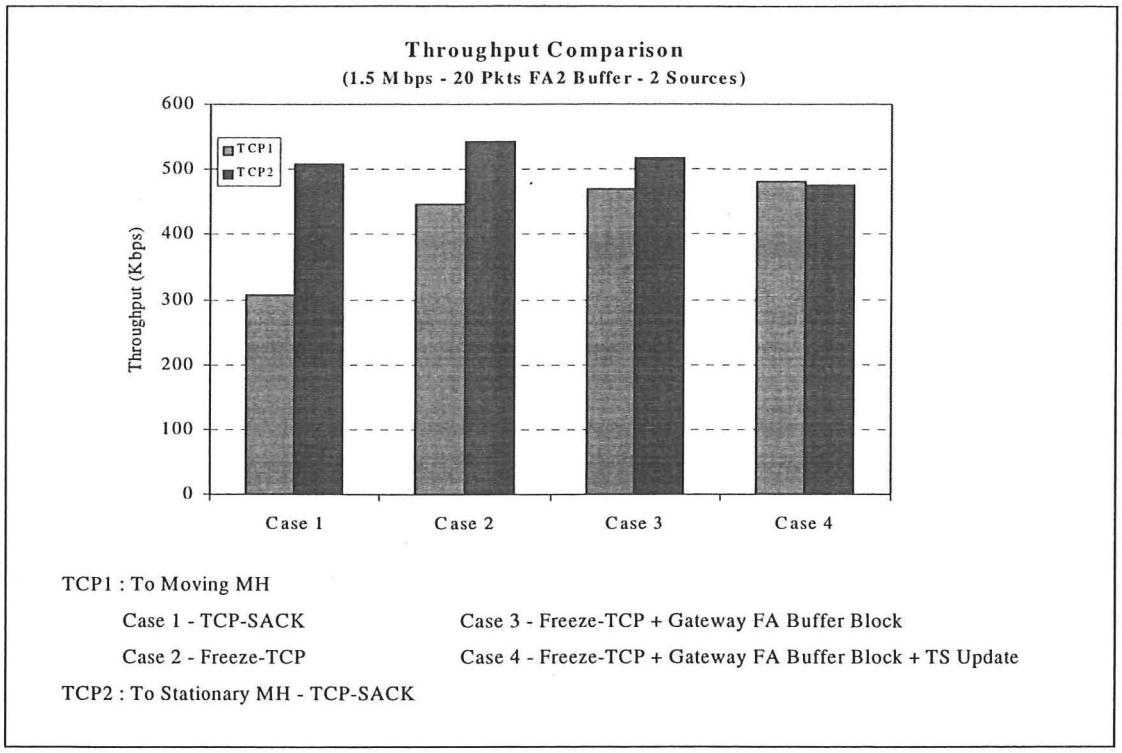

Figure 6.4.1-1: Effect of window freezing for handoffs with 20KB buffers and two background sources.

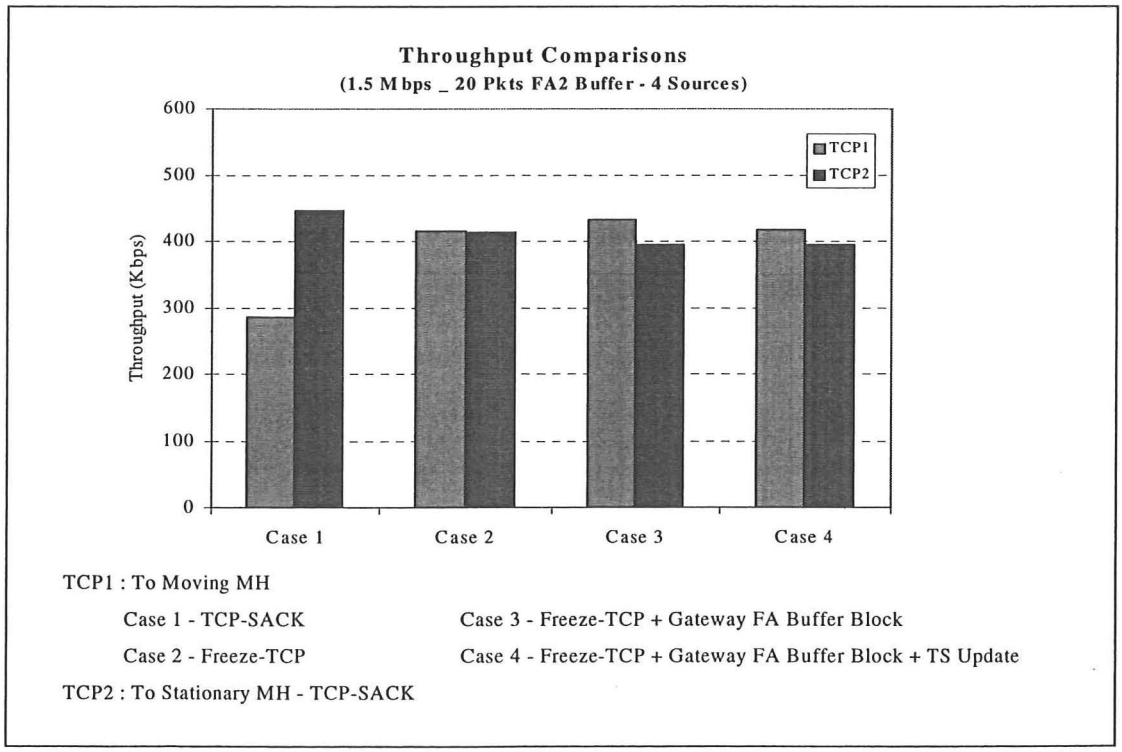

Figure 6.4.1-2: Effect of window freezing for handoffs with 20KB buffers and four background sources.

TCP-SACK

The congestion window plot of the mobile TCP-SACK connection exemplified in Figure 6.4.1-3 shows that on several occasions it was penalised by the large recovery time following timeouts when the MH entered the more loaded cell and long recovery time.

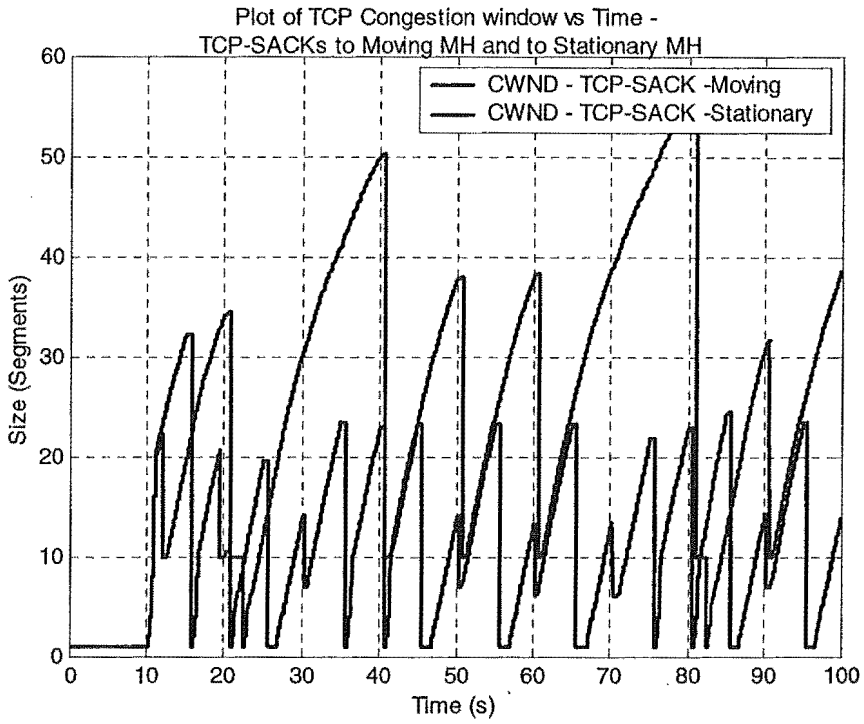

Figure 6.4.1-3: TCP Congestion window for TCP-SACKs when two background sources are used.

Freeze-TCP

Freeze-TCP not only had significant throughput improvements of more than 40% over the case when TCP-SACK is used for the mobile connection but it also allowed a small improvement of almost 7% to the competing stationary TCP-SACK connection when the lower background traffic level is used. Analysis of the results in Table A-3 shows that the number of timeouts and retransmissions suffered by the TCP-SACK competing with Freeze-TCP is reduced in relation to that observed when TCP-SACK is used in the mobile connection This is attributed to the fact that Freeze-TCP freezes the sender prior to the handoff and hence the competing TCP is free to send data

without competition from the connection using Freeze-TCP during this period. In this scenario halving the load in the two background sources was also seen to provide similar performance gain to the competing TCP-SACK when using Freeze-TCP and comparing to the case when TCP-SACK is used for the mobile connection.

Figure 6.4.1-4 shows an example of the congestion window of the mobile connection with Freeze-TCP, as well as the congestion window of the competing connection. It can be seen that, although Freeze-TCP still reduces its window due to Fast Retransmits, it is able to avoid the need for timeouts. Hence significant performance improvement over TCP-SACK is achieved. Note also that the TCP-SACK competing with Freeze-TCP is still able to grow its window to the maximum allowed value (the advertised window value) on several occasions.

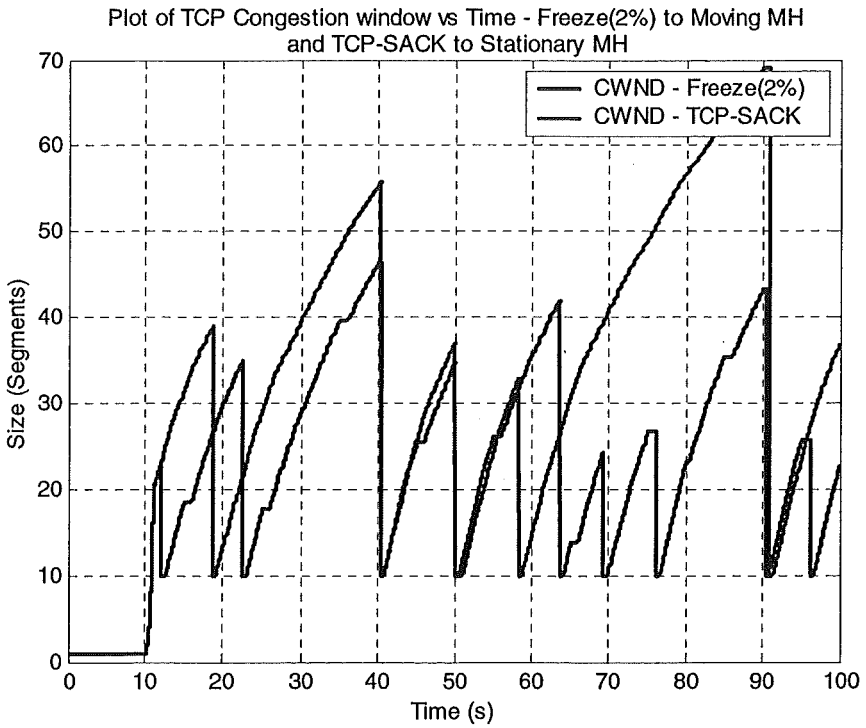

Figure 6.4.1-4: TCP Congestion window of Freeze-TCP and competing TCP-SACK when two background sources are used.

Freeze + Gateway Blocking + TS Echo Update

The buffer scheme with timestamp echo updates was able to perform almost 8% better on average than Freeze-TCP when two background sources are used, since it is able to

freeze the sender later by successfully using a smaller warning threshold than Freeze-TCP. This happened in spite of the fact that, in contrast with Freeze-TCP, this window-freezing scheme suffered a small number of timeouts. This situation is exemplified in Figure 6.4.1-5.

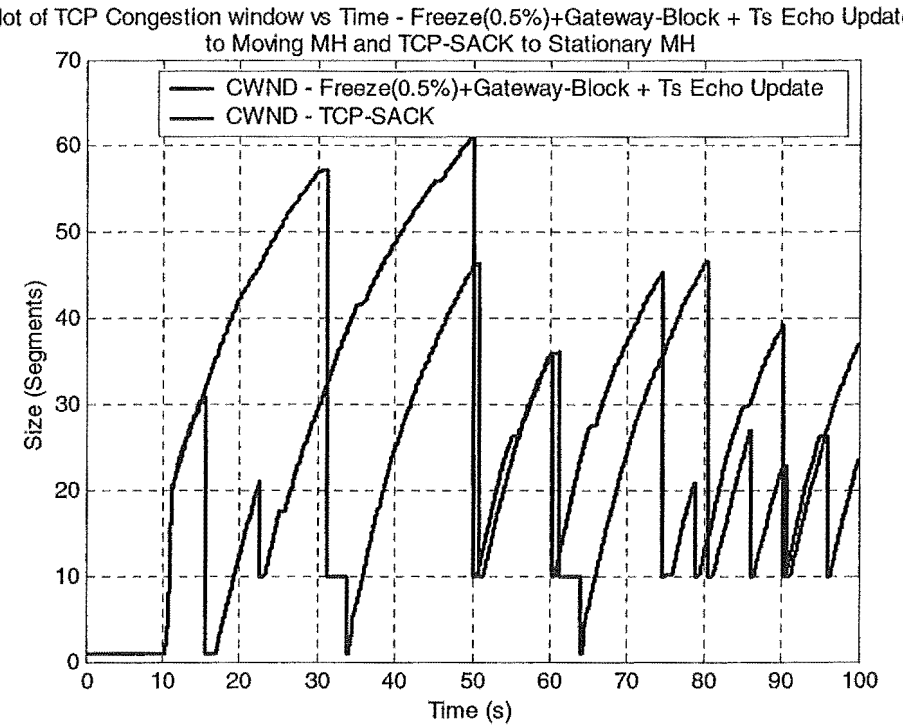

Figure 6.4.1-5: Congestion window of Freeze-TCP with Gateway FA buffering and timestamp echo update using two background sources.

The burst of data from the Gateway FA leads to increased losses at the new BS when compared to Freeze-TCP. This is due to the smaller delay between the sending buffer agent at the Gateway FA and the MH when compared to the delay between sender and MH. The higher bandwidth between the Gateway FA and the BSs than the bandwidth between the sender and the Gateway FA also means that bursts of packets from the Gateway FA may have a more detrimental effect than bursts from the TCP sender.

Therefore the buffer scheme without timestamp updates was seen to perform equivalently to Freeze-TCP under this scenario upon statistical analysis of the results. Furthermore, when the congestion level was increased by using four background sources rather than two, Figure 6.4.1-2 and further statistical analysis show that, regarding the mobile connection, the performance of all window freezing schemes is

equivalent and the improvement over using TCP-SACK for the mobile connection is similar to when the lower background traffic level is used. Now the buffer schemes suffer from more losses due to the excessive burst of data into the more congested BS buffer and hence no significant improvement over Freeze-TCP is observed.

As could be expected, the use of a more aggressive scheme, which uses buffering to freeze the sender later and recover faster from multiple losses brought performance degradation to the competing connection. Hence the TCP-SACK of the fixed connection competing with the scheme with buffering and timestamp echo updates performed almost 7% worse than the competing TCP-SACK when TCP-SACK is also used for the mobile connection.

The increased congestion caused by the addition of two more background sources makes all the window freezing schemes more aggressive since they have enhanced ability to maintain a high transmission rate. As a result, the schemes using buffering at the Gateway FA brought a performance degradation of approximately 10% to the competing TCP-SACK when compared to the case where two TCP-SACKs are used.

Discussion

Figure 6.4.1-6 shows a comparison of typical receiver traces of the mobile connection using TCP-SACK, Freeze-TCP and Freeze-TCP with Gateway blocking and timestamp echo update for the case when two background sources are used. It can be seen that the window freezing schemes successfully avoided extended idle times during handoffs. The window freezing connection with buffer blocking and timestamp echo updates further reduces idle times in comparison to Freeze-TCP in a small number of occasions. This situation happens when a timeout does occur. The use of the timestamp echo update then allows for faster recovery.

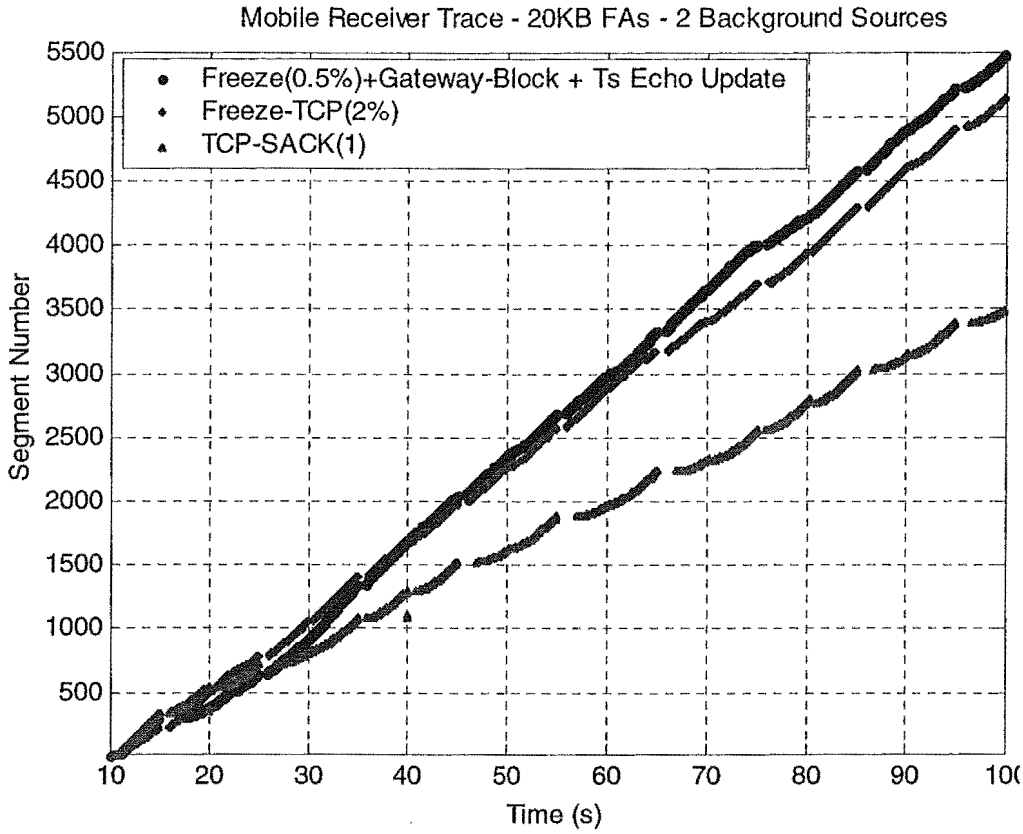

Figure 6.4.1-6: Receiver trace of mobile connection of TCP window freezing schemes and unmodified TCP-SACK.

Therefore, in the equal buffers scenarios tested, it was observed that the window freezing mechanisms had the effect of bringing the mobile connection average received throughput close to that obtained by the stationary connection. A much higher throughput improvement was obtained by the mobile connection than the degradation seen by the competing unmodified connection due to the window freezing mechanisms, which raised the overall performance.

The limited BS buffer size, the presence of competing traffic and the reduced ability to absorb the burst of data lead to some congestion losses when buffering was used. However, the buffer size used is large enough to limit such losses and hence the performance of this scheme is not significantly penalised by such bursts from the Gateway FA. This issue will be discussed in more detail when we consider the buffer capacity mismatch simulations.

6.4.2. Buffer Capacity Mismatch Scenario

Figures 6.4.2-1 and 6.4.2-2 show the simulation results for the case where the BSs have different buffer sizes, corresponding to a scenario where the MH can move into a cell with a higher congestion level than the previous cell.

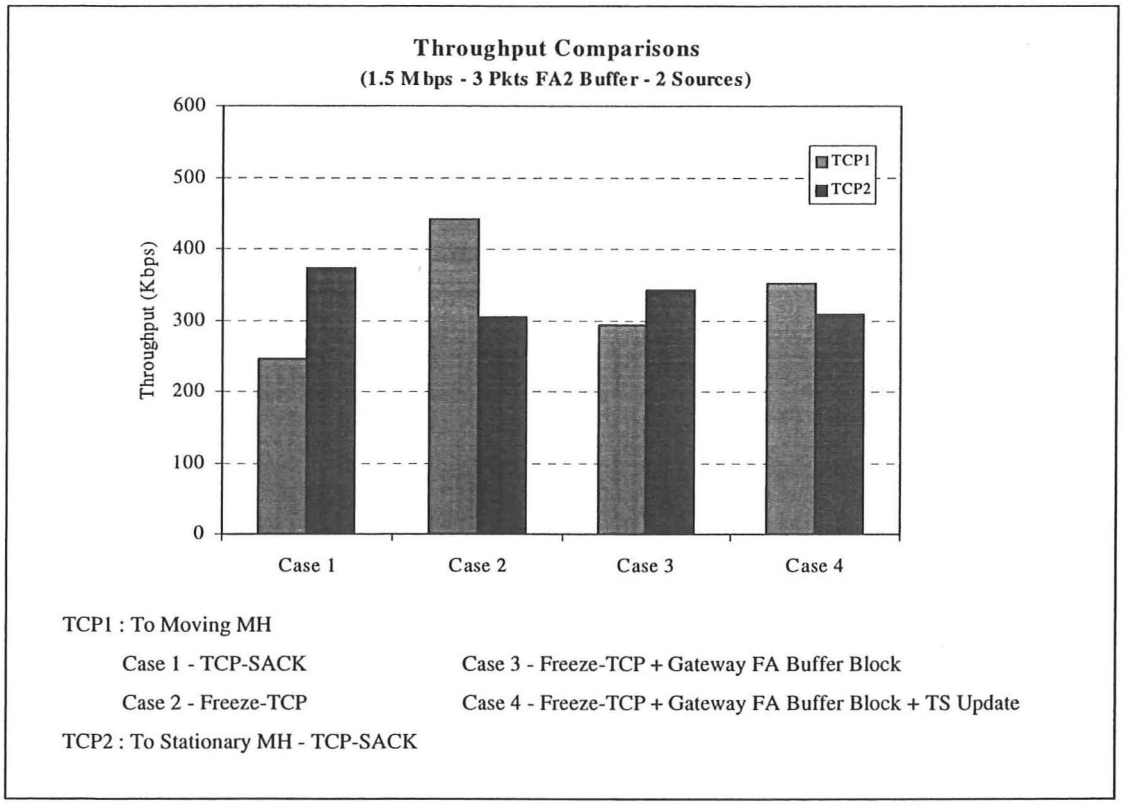

Figure 6.4.2-1: Effect of window freezing mechanisms on TCP-SACK for mismatched load handoffs using two background sources.

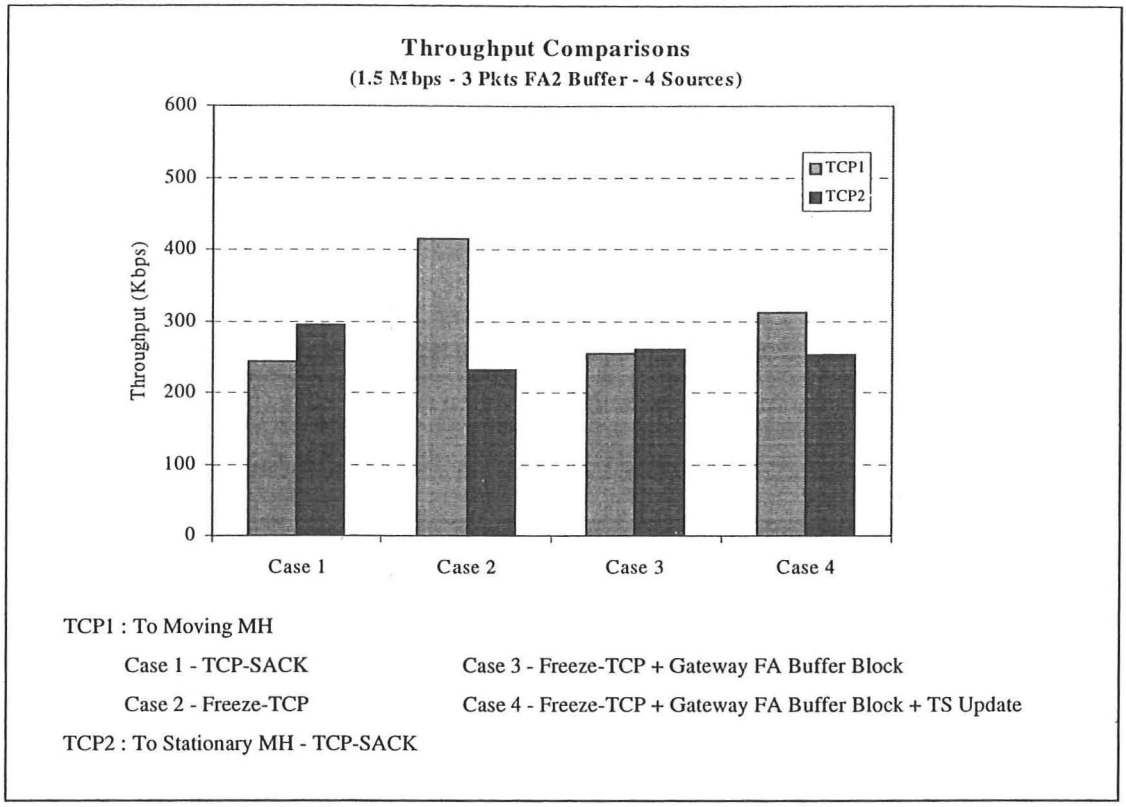

Figure 6.4.2-2: Effect of window freezing mechanisms on TCP-SACK for mismatched load handoffs using four background sources.

TCP-SACK

As expected, the increased congestion brought about by the smaller FA2 buffer size reduced the performance of TCP-SACK to the mobile connection in comparison to the case where 20KB buffers were used. In this scenario, the reduction of the TCP-SACK congestion window is more problematic than when large buffers were available, since now a timeout penalises a connection more severely. When a connection suffers a timeout, the competing TCP connection can utilise the bandwidth used by the other connection, which can make the connection that timed out unable to restart transmission for extended periods [Sarolahti 2000].

A situation where this happens can be seen between times 35 seconds and 50 seconds into the simulation shown in Figure 6.4.2-3. It can be seen that after the moving MH enters the congested cell at times 35 seconds and 45 seconds, it reduces its congestion window and is unable to have significant growth of the congestion window until it enters the less congested cell following the next handoff. The competing TCP-SACK,

on the other hand, was able to take advantage of this situation and grow its congestion window further.

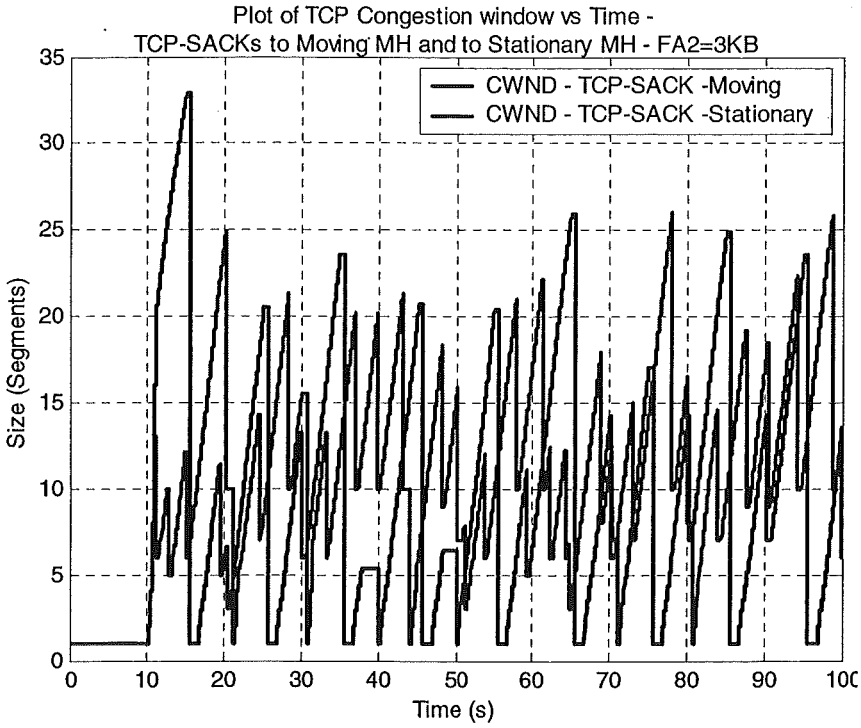

Figure 6.4.2-3: Performance degradation of mobile TCP-SACK connection when different buffer sizes at the BSs and two background sources are used.

Freeze-TCP

Although Freeze-TCP was affected by the congestion level due to the number of background sources, the change in FA2 buffer size did not significantly degrade its throughput when comparing the obtained throughputs for different buffer sizes and a given number of sources in the cases tested. This means that when buffer overflow due to the burst following a reconnection occurred, Freeze-TCP was still able to recover from these losses efficiently and the amount of losses due to inadequate congestion window following a handoff is limited.

Figure 6.4.2-4 shows an example of the congestion window of the Freeze-TCP connection and the competing TCP-SACK when four background sources are used. It can be seen that Freeze-TCP was still able to avoid timeouts, suffering only approximately one timeout on average when four sources are used (Table A-12).

Besides this, the figure shows that the congestion window of the window freezing connection is also reduced on several occasions that are unrelated to handoffs[1], due to the small buffer size available at FA2. These however are mainly due to fast retransmissions and the TCP sender is still able to grow its congestion window even while in the more loaded cell.

The Figure also show several occasions where the MH enters the coverage area of the cell with the smaller buffer capacity, the congestion window is successfully restarted at the level used in the cell with a smaller congestion level and allowed to start growing again. It is seen, however, that a fast retransmission soon follows. This is due to the inadequate level of the CWND for the more congested cell buffer. Hence, although the old congestion window leads to losses, it is not necessary to reduce the CWND to its initial value and only a fast retransmission is needed for recovery.

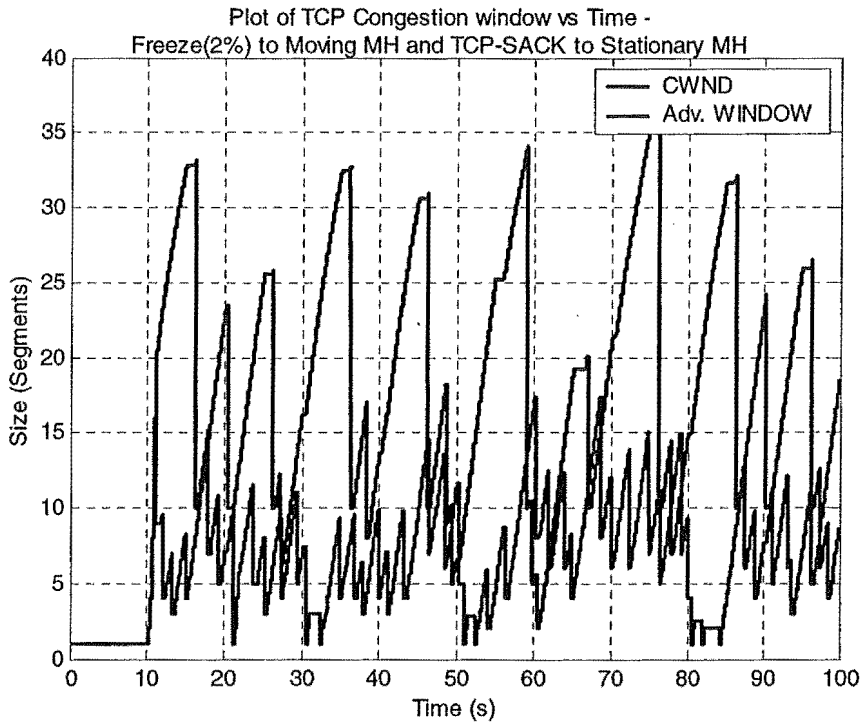

Figure 6.4.2-4: Congestion window of Freeze-TCP and competing TCP-SACK when four background sources are used in load mismatch scenario.

[1] As explained in Chapter 5, handoffs occur every five seconds, starting at 15s, in the simulations.

Since Freeze-TCP was able to avoid excessive throughput degradation due to timeouts in the cases of load mismatch, as opposed to TCP-SACK, and hence was able to compete better with the stationary connection, a large benefit was seen in this scenario from using Freeze-TCP instead of TCP-SACK in the mobile connection. Hence, for the mobile connection, throughput improvements of approximately 80% and 70% were achieved by using Freeze-TCP instead of TCP-SACK when two and four background sources were present, respectively.

The problem of burst overload of the new BS does not seriously affect Freeze-TCP in our environment due to the larger buffering capacity at the network in the case of Freeze-TCP. This is due to the larger delay between the sender and receiver than the delay between Gateway FA and receiver. Also, packets from the Freeze-TCP sender burst following a reconnection can be stored at the Gateway FA, instead of being sent directly to the congested new BS. Furthermore, the presence of a bottleneck link of 1.5Mbps at the wired network and the fact that a faster 10Mbps link exists between the Gateway FA and the BSs, means that packets arrive at the BSs spaced according to the speed of the slower 1.5Mbps link [Stevens 1994]. Therefore, sending packets from the TCP sender will allow more time for the queue at the congested BS to drain than if the Gateway FA sends packets directly according to the faster link speed. Hence the burst following a reconnection does not seriously affect the new BS.

The average throughput performance benefit of Freeze-TCP is not as high in the case where four background sources are used as in the case where two background sources are used since in the former case more congestion exists and hence Freeze-TCP is forced to reduce its window more severely.

The previous discussion shows that, when a large congestion mismatch between the new and old cell buffers occurs, Freeze-TCP becomes more aggressive against competing connections than in the case where the old and new BS buffers have similar available storage capacities. Hence, the performance penalty suffered by the TCP-SACK competing with Freeze-TCP is increased to approximately 20% as can be seen in Figures 6.4.2-1 and 6.4.2-2. However, the congestion window plots showed that freezing the TCP sender avoided the mobile connection from being disadvantaged

by the presence of disconnections in scenarios where the mobile connection is further penalised by limited resources available.

Freeze + Gateway Blocking

As opposed to the equal buffer sizes scenario, the buffering schemes are seen to perform significantly worse than Freeze-TCP in the scenarios with load mismatch. This is caused by the overload of the new cells by the burst of data from the buffer agent to the already congested BS. An example of this situation is shown in Figure 6.4.2-5.

In this case the MH moves into the more congested BS (FA2) coverage area at time 75 seconds. When the handoff is completed, the Gateway FA sends the packets buffered prior to the handoff to the new BS. The figure shows that the burst of buffered packets following the handoff to the BS with a higher congestion level causes the MH to receive only a small number of packets sent by the Gateway FA. It also shows that a number of packets are received by the new BS (FA2) but not received by the MH. Examination of the simulation trace file showed that these packets are marked as arriving at FA2 and then dropped at FA2 due to buffer overflow. Another fact shown by this figure is that the first packets sent in the burst following the reconnection get to the MH successfully.

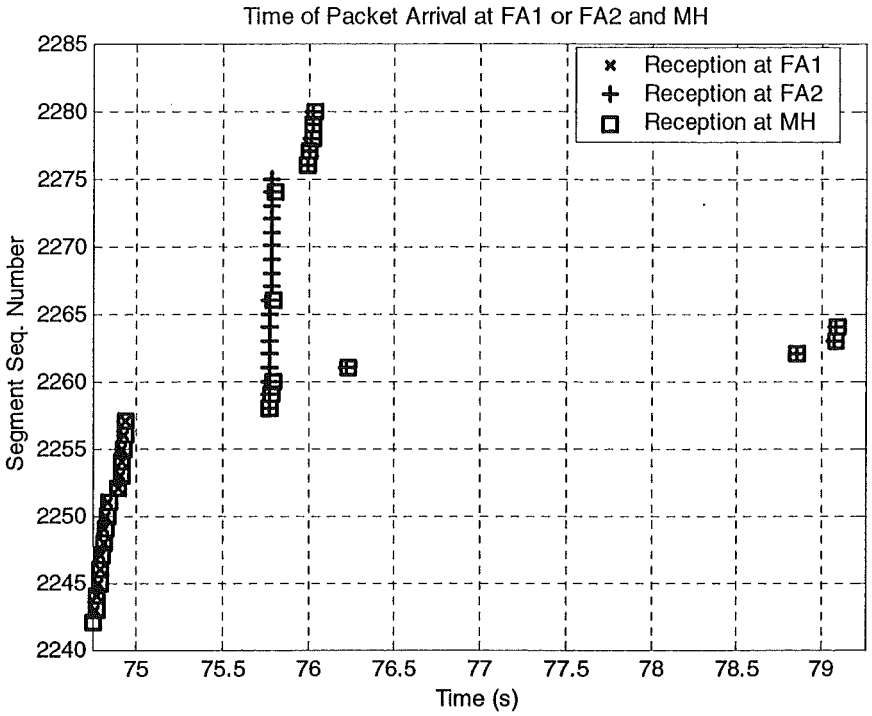

Figure 6.4.2-5: Loss of segments at congested BS due to burst from Gateway FA in four background sources and 3 packets FA2 buffer size scenario.

Since multiple packets are lost, a timeout will be required to restart the sender. The ACKs for the packets buffered at the Gateway FA that do get to the mobile and the TR-ACK sent by the MH to warn the sender about the MH reconnection will inflate the RTO since they ACK new data at the sender, as described in Section 4.4. Therefore a long waiting time will be required to restart the sender. The problem in this case is more significant than the problem noted in Section 6.3 due to inflation of the RTO caused by the TR-ACK sent by the MH acknowledging new data. This is because now more ACKs for new data are received and hence more inflated samples are used to update the RTO.

The sender trace of Figure 6.4.2-6 shows that the ACKs for the buffered packets and the TR-ACK sent by the MH take the sender out of persist mode. Since packets are lost, the new packets sent after the sender leaves persist mode generate DUPACKs. The figure shows that the sender remains idle waiting for the timeout for approximately 2.5 seconds after the fast retransmission. This is in contrast with the fact that the RTT before the handoff was less than a second. The bursts of data following MH reconnections and the RTO inflation lead to several timeouts and excessive idle time throughout the connection, as shown in Figure 6.4.2-7.

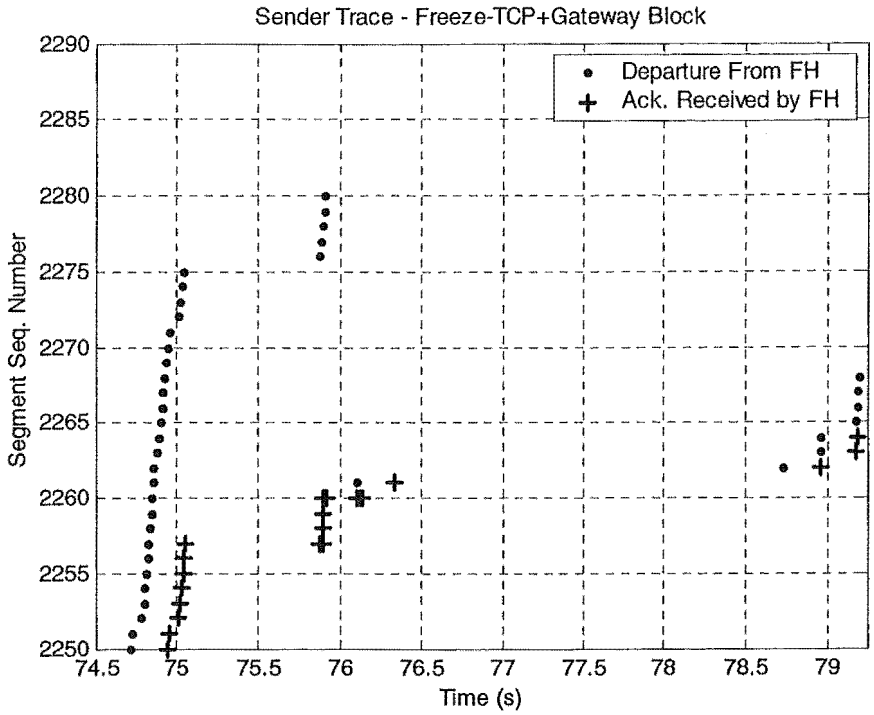

Figure 6.4.2-6: Sender idle time due to RTO inflation caused by Gateway FA burst to congested BS with four background sources used.

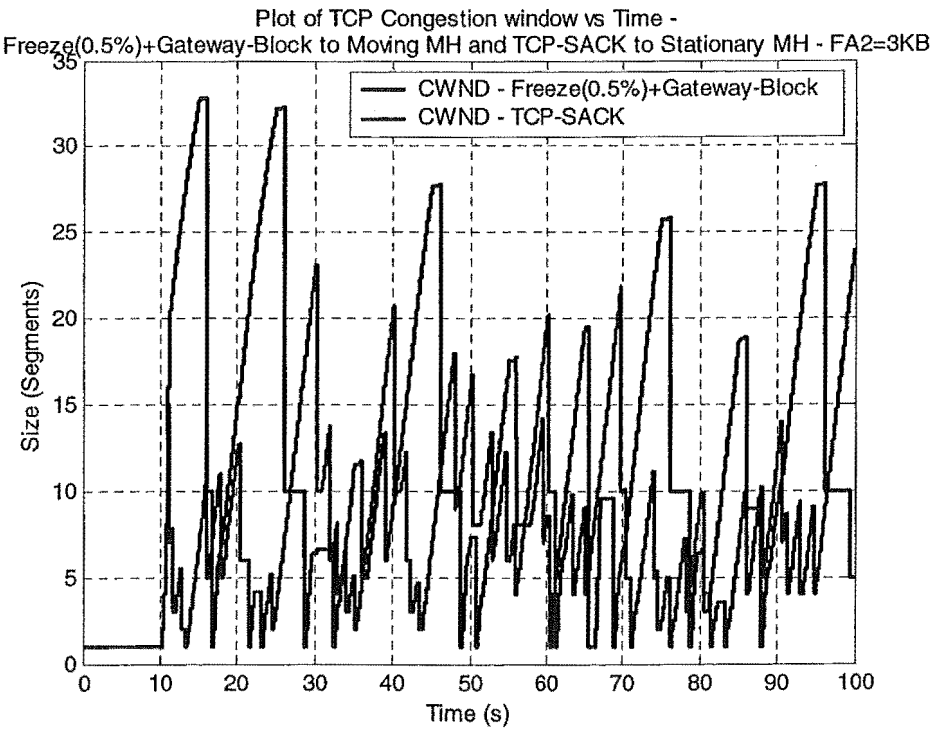

Figure 6.4.2-7: Congestion window of Freeze-TCP with Gateway blocking and of competing TCP-SACK using four background sources in load mismatch scenario.

Even though the Gateway FA buffer blocking without timestamp update scheme performed significantly worse than Freeze-TCP when two background sources were used, it still performed almost 20% better than using TCP-SACK for the mobile connection. When the congestion level is further increased by adding another two background sources, however, the performance of the buffer blocking scheme without timestamp update is comparable to that of using TCP-SACK in the mobile connection. It is noted that the problem of overloading the new BS following a handoff can also occur if the window freezing mechanism is used with the proactive buffering and post-handoff redirection policy, as in [Onoe et al 2001].

The increased congestion caused by the burst of data to the already congested BS also contributes to the worsening of the competing TCP-SACK connection. The burst not only causes packets from the window freezing connection to be lost, but it also leads to the aggravation of congestion in the network, adversely affecting the competing TCP-SACK connection as well. The occupation of the buffer space available with packets of the window freezing scheme increases the unfairness against the competing TCP-SACK. Therefore, even though the average performance of the connection with window freezing and buffering capabilities did not improve, the performance of the competing TCP-SACK is still approximately 10% worse than when TCP-SACK is used to the mobile connection.

Freeze + Gateway Blocking + TS Echo Update

The addition of the timestamp echo update mechanism described in Section 4.5 is seen to significantly improve the performance of the buffer scheme in the cases of load mismatch (Figures 6.4.2-1 and 6.4.2-2). Such improvement is due to the lower time required for the timeout to occur since the RTO inflation is avoided through the use of the timestamp echo update mechanism. This can be seen by comparing the congestion window plots in Figure 6.4.2-7 and Figure 6.4.2-8. Note that in the latter case, although fast retransmissions followed by timeouts still occur, the time to restart the sender is not as high as in the former case.

Despite the improvement in average throughput, the average number of timeouts and retransmissions shown in Table A-12 is still equivalent to that seen when no timestamp echo update is used. This shows that the addition of the timestamp echo update scheme reduced the sender idle time and hence allowed increased throughput.

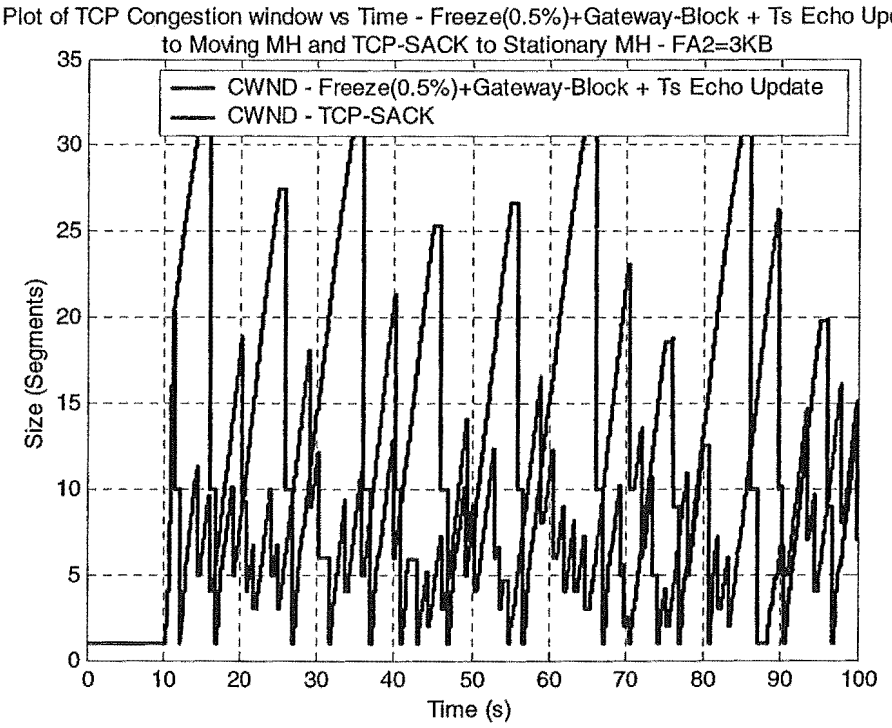

Figure 6.4.2-8: Congestion window of Freeze-TCP with buffer blocking and timestamp echo updates and competing TCP-SACK when four background sources are used.

The registration request announcing the new BS is handled before sending the buffered packets at the Gateway FA so that when a handoff occurs the buffered packets are sent to the correct BS. Hence, the registration reply is sent to the MH before the buffered packets. Therefore the timestamp echo is updated before the buffered packets arrive and, when the buffered packets arrive, the ACKs for these packets will have updated echo timestamps that do not include the disconnection length.

The buffered packets were generated before the disconnection and the disconnection duration is such that the difference between the reconnection time and the time when

the last ZWA was sent is greater than the inter-segment send time. Hence, the timestamps of the buffered packets are smaller than the updated timestamp echo (ts_echo_) variable. Therefore the timestamp echo bug fix [Wright and Stevens 1995] (see Section 2.1.1.2) does not allow for the timestamps of these packets to be used to update the timestamp echo variable. This avoids the problem of RTO inflation due to the ACKs for these packets. However, this also means that the buffered packets are all sent with the same timestamp echo.

For more accurate RTT estimation it would be better to perform the same updates for these buffered packets so that the difference in time between sending the segments that were buffered is also taken into account when calculating the RTT. This avoids the risk of creating spurious timeouts when a timeout is not needed following the reconnection but may retard the occurrence of the timeout when a timeout is needed due to losses during the disconnection. Note, however, that since the ACKs for the buffered packets unfreeze the sender, the segments sent after the sender is unfrozen will update the timestamp echo variable according to the mechanism in [Wright and Stevens 1995] and provide precise RTT estimate.

Not updating the timestamp echo using the timestamp of buffered segments was not seen to cause performance problems in our simulations, since the average number of timeouts of the window freezing connection with buffering with and without the timestamp echo update mechanism was practically unchanged in the scenarios simulated (see tables A-9 and A-12).

The discussion above explained the reason for the significant performance improvement over the case where the proposed timestamp update mechanism was not added to the buffering scheme. Approximately 40% and almost 30% performance improvement is achieved over using TCP-SACK in the mobile connection in this scenario, therefore providing a significant performance improvement over the case where the timestamp echo is not updated after the reconnection detection.

Although the performance gain of the scheme using buffering and timestamp echo updates is reduced when the congestion level is increased, substantial performance

gain over using TCP-SACK for the mobile connection is still achieved. This contrasts to the case where timestamps updates are not used.

6.4.3. Summary

In this section it was observed that TCP-SACK was severely penalised by handoff-induced timeouts when a competing stationary connection is present. This was particularly serious in cases of heavy congestions since the mobile TCP-SACK connection was unable to reclaim resources obtained by the competing stationary connection. Freeze-TCP on the other hand was seen to be significantly less affected by this problem and was able in most cases to recover from handoffs through fast retransmissions.

Although the use of Gateway FA buffer blocking was seen to be beneficial in light congestion cases, it proved to be a problem when heavy congestion existed. This was due to the use of timestamps from buffered packets in the RTO calculation and the loss of several packets due to excessive burst from the Gateway FA. The use of the timestamp echo update mechanism was hence seen to be particularly important.

6.5. Burst Control

In the previous section, it was observed that, in our simulation environment, the buffer schemes suffered a performance penalty due to the excessive burst of data sent by the buffer agent upon reconnection detection. It was also observed that Freeze-TCP was able to perform better due to the slower link speed and larger link delay between the sender and the Gateway FA when compared to the bandwidth and delay between Gateway FA and MH in the conditions tested.

In this section, the burst limit mechanism proposed in Section 4.6.5.1 is added to our previous scheme of coupling Gateway FA buffering with timestamp echo updates and its performance is compared to that of TCP-SACK, Freeze-TCP and Freeze-TCP with Gateway FA buffering under various scenarios. As discussed in Chapter 5, a burst limit of two packets is used in these simulations.

The next subsections present and discuss the simulation results for the configuration used in the previous section and also for the case where all links in the network have equal speeds. The latter configuration is useful to analyse and compare the performance of Freeze-TCP and the buffering mechanisms in the presence of equally large link speeds.

6.5.1. Link Speed Mismatch Scenarios

This subsection analyses the performance obtained by our window-freezing scheme with timestamp echo updates and burst limiting mechanisms in the configuration used in the last section. Hence, all wired links between the sources and the Gateway FA have a bandwidth of 1.5Mbps while the links between the Gateway FA and the BSs are 10Mbps.

First the case where all the BS buffers have capacity for 20 packets and two background sources are used is presented. Then the case where buffer capacity mismatch exists between the BSs is presented using two and four background sources.

Equal Buffer Sizes Simulation

When large buffers are used in the BSs, the buffering agent is still able to perform appropriately by avoiding the occurrence of excessive timeouts even when the entire Gateway FA buffer contents is sent to the new BS, as seen in the previous section. Hence no significant advantage or performance degradation was seen by adding the burst limit control and timestamp echo updates mechanisms to the base Freeze-TCP with buffer blocking upon disconnection prediction.

Load Mismatch Scenarios

As opposed to when the old and new cells have similar, large, buffers, the addition of the burst limit scheme to the buffer agent was seen to bring significant performance improvement over the case when all the buffer contents are sent upon reconnection

detection in situations where significant capacity mismatch exists between the old and new BS buffers. This is shown in Figure 6.5.1-1 and Figure 6.5.1-2.

In the first figure it can be seen that when two background sources were used, the performance is now equivalent to that of Freeze-TCP. Analysis of the results in Table A-9 shows that the average number of timeouts seen during the simulations is significantly reduced in comparison to when the entire buffer contents are sent to the new BS upon reconnection. The burst control was therefore useful in this case to reduce the congestion at the more heavily loaded BS. Hence the idle time is reduced and the congestion window is kept larger, leading to the increased throughput observed.

The burst limit scheme with timestamp echo updates as expected still performed significantly better than when burst control and timestamp updates are not used when the level of congestion is increased by adding two extra background traffic sources. This can be seen in Figure 6.5.1-2. It also performed approximately 4% better than Freeze-TCP. Now the benefit of using the timestamp echo update is higher than in the case of less congestion and hence using this mechanism together with the burst control mechanism provides enhanced performance to the window freezing connection when timeouts following a MH reconnection still occur due to buffer overflow.

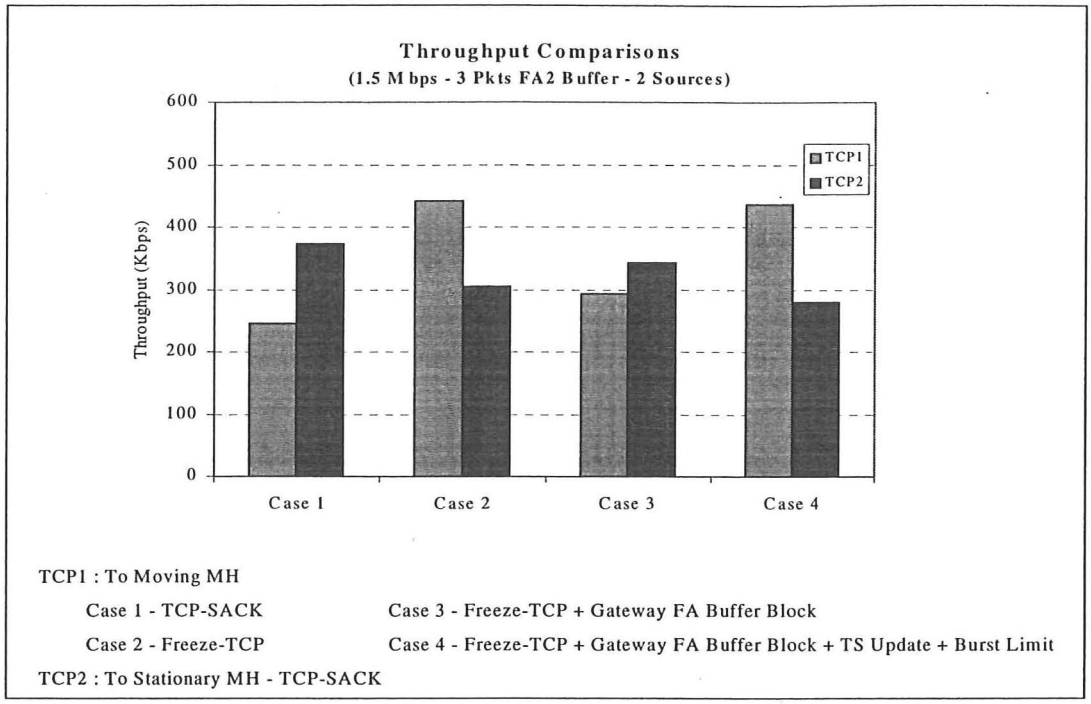

Figure 6.5.1-1: Throughput comparisons including burst limiting scheme in buffer mismatch scenario when using wired bottleneck link of 1.5Mbps and two background sources.

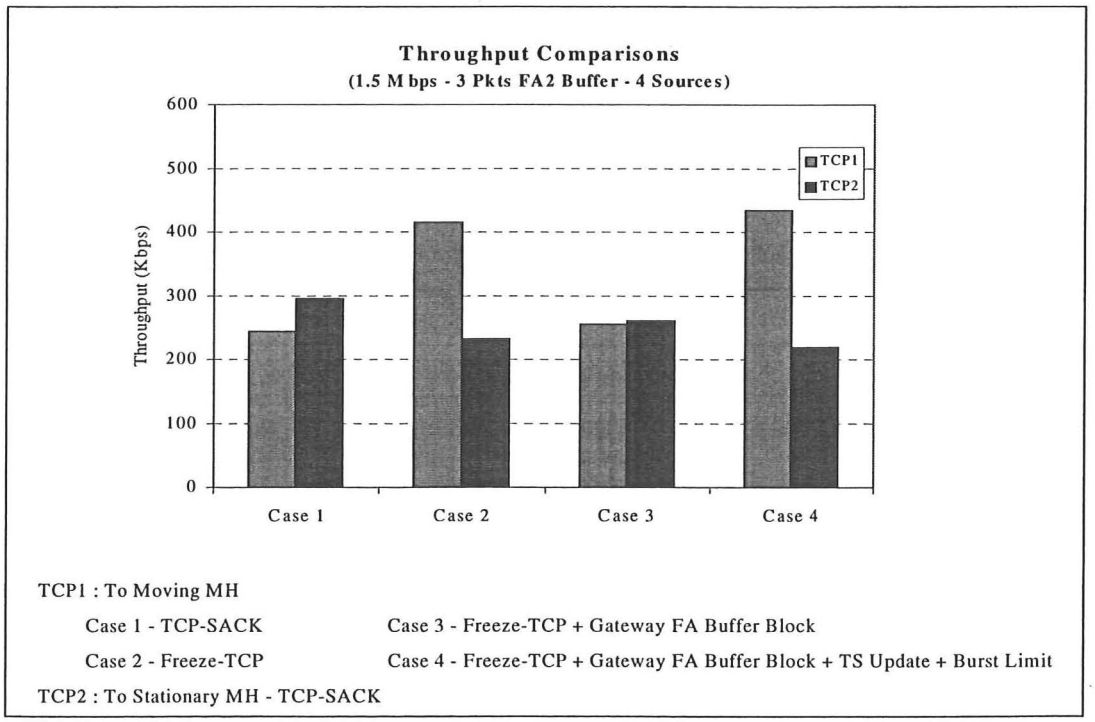

Figure 6.5.1-2: Throughput comparisons including burst limiting scheme in buffer mismatch scenario when using wired bottleneck link of 1.5Mbps and four background sources.

The presence of background traffic and a competing TCP connection mean that sending two packets to the buffer with capacity for three packets will still lead to losses on various occasions. Hence the inclusion of the timestamp echo update is also beneficial and avoids excessive performance degradation of the overall proposed scheme.

The amount of losses, however, is significantly reduced in comparison to when burst limit is not used. This helps in enabling the recovery through a fast retransmission to be successful rather than requiring the need for a timeout. Hence only approximately one timeout on average was observed during the simulations using a maximum burst of two packets and four background sources in the case where the BS buffers have different capacities. This is similar to the number of timeouts observed when using Freeze-TCP in these conditions.

The objective of sending more than one buffered packet when a single ACK is received is to allow for a simple scheme that will attempt to clear the buffer at the Gateway FA, which may be required to handle several connections at a time, as soon as possible. Hence it has the disadvantage of still aggravating the congestion in cases where the new BS is heavily congested. This, however, is still better than sending the entire buffer contents to the new BS, as is done in the buffer blocking and redirection mechanism to deal with handoffs in [Onoe et al 2001].

It is observed that the buffer agent in our scheme also needs to absorb the burst from the TCP sender since it is located at the Gateway FA rather than at BSs. Therefore it is an advantage that the buffer agent attempts to send packets at a high rate when the reconnection is detected so that it can accommodate packets arriving from the TCP sender when it is unfrozen. The buffer agent must, however, limit its transmission rate to avoid overloading the new BS buffer, as previously discussed.

The statistical test showed that a comparison of the competing TCP-SACK results for the cases when the window freezing with timestamp echo updates and burst limit or Freeze-TCP were used in the mobile connection were not conclusive due to the variation of results among simulation runs. However, a performance degradation seen

by the TCP-SACK connection competing with the scheme with burst limit in comparison to the performance obtained by the TCP-SACK competing with Freeze-TCP, such as that observed by comparing the throughput averages obtained is to be expected. This, as in other cases where resources are limited, is due to the increased ability of the connection with burst limit to maintain a higher congestion window and hence better use the resources available.

6.5.2. Uniform Link Speeds Scenarios

Previous simulations showed scenarios where Freeze-TCP was not affected by the burst of data following a reconnection while the buffering mechanisms without burst control were adversely affected by the excessive burst of data, particularly in cases of large load mismatch between the old and new cells.

In this section, the same link speed of 10Mbps is used in all the links in the wired portion of the network. Therefore, we observe whether or not Freeze-TCP is still able to avoid losses following a burst from the sender after a reconnection in this scenario. The performance of Freeze-TCP is compared to that of the Freeze-TCP with buffer blocking as well as with the performance of the proposed Freeze-TCP with buffer blocking enhanced with timestamp echo updates and burst limit control. The unmodified TCP-SACK implementation is again used as a baseline for comparison.

Equal Buffer Sizes Simulation

In the test of the burst limit control in this scenario, the variability of the simulation results do not allow for accurate conclusions to be drawn about the difference in results among the window freezing schemes. Hence the lower throughput of the burst control scheme in comparison to Freeze-TCP seen in Table A-13 is not statistically significant due to the variability in the test results used to obtain the throughput averages in Table A-13.

The analysis of the effect of the window freezing connections on the TCP-SACK connection to the stationary MH in the current configuration shows that the buffer

blocking with burst control and timestamp echo updates reduced the throughput of the connection to the stationary TCP connection by approximately 7% (Table A-13) in relation to when Freeze-TCP is used in the mobile connection. This indicates the achievement of a slightly better performance of the scheme using burst control, which as a consequence had the more detrimental effect on the competing TCP-SACK connection, due to the limited network capacity available and the ability of the former connection to maintain a high transmission rate. The average throughput results in the present configuration are summarised in Figure 6.5.2-1.

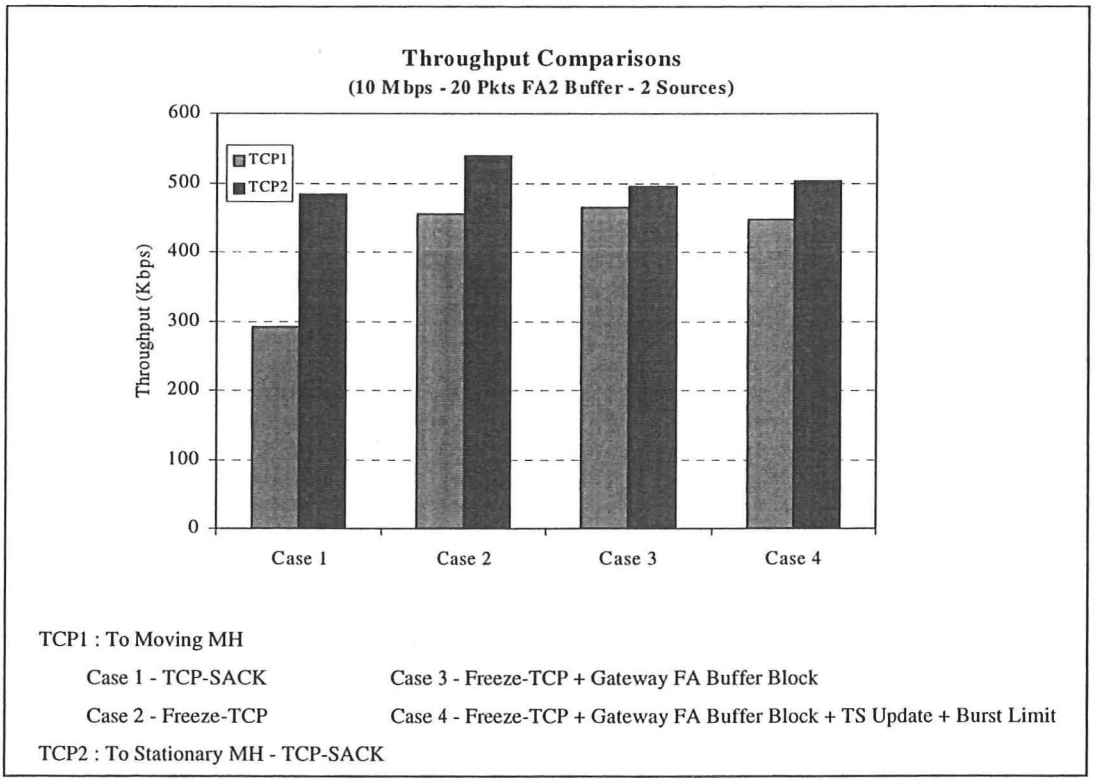

Figure 6.5.2-1: Throughput of window freezing schemes using 10Mbps wired link speeds, two background sources and BS buffers of equal sizes.

The link speed in the present section is uniform and considerably higher than when a bottleneck link of 1.5Mbps is present in the wired network. Hence the packets coming from the TCP sender arrive closer to each other at the BSs than when a slower link exists in the wired network, leading to more severe congestion at the FA2 buffer.

In the scenarios of Section 6.5.1, where the bottleneck link in the wired network was 1.5Mbps, the speed mismatch at the interface of the wired and wireless networks was

not so serious. Note that the BS can still suffer from congestion even when a 1.5Mbps link exists in the wired network and the wireless link bandwidth is 2Mbps. This is due to the contention problem caused by the IEEE MAC 802.11 protocol used in the simulations.

When the BSs had larger buffers with capacity for 20 packets, the performance degradation caused by having a faster wired network interfacing to the slow wireless link was not seen to be significant due to the higher buffering capacity at the BSs.

Load Mismatch Scenarios

Figure 6.5.2-2 and Figure 6.5.2-3 show a summary of the results obtained when two and four background sources are used, respectively, in the scenario where all wired links are 10Mbps and the BSs have unequal buffer sizes of 20 and 3 packets. The rest of this section discusses the results obtained.

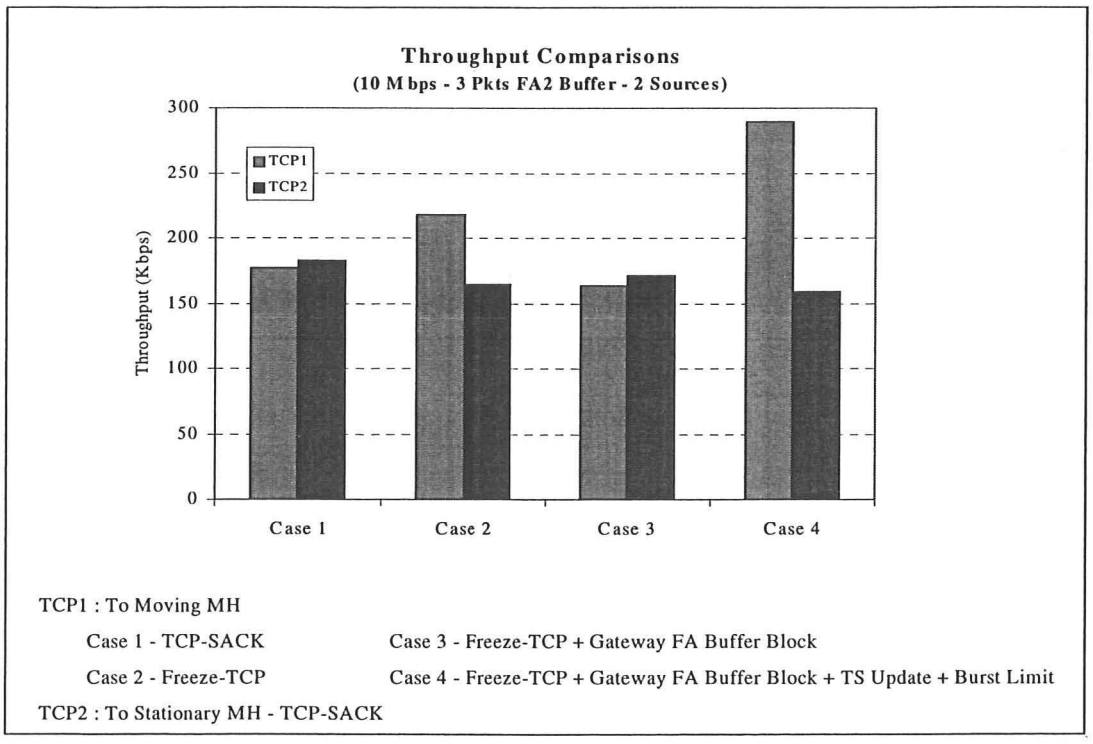

Figure 6.5.2-2: Throughput comparison of TCP-SACK and window freezing mechanisms under load mismatch and uniform link speeds using two background sources.

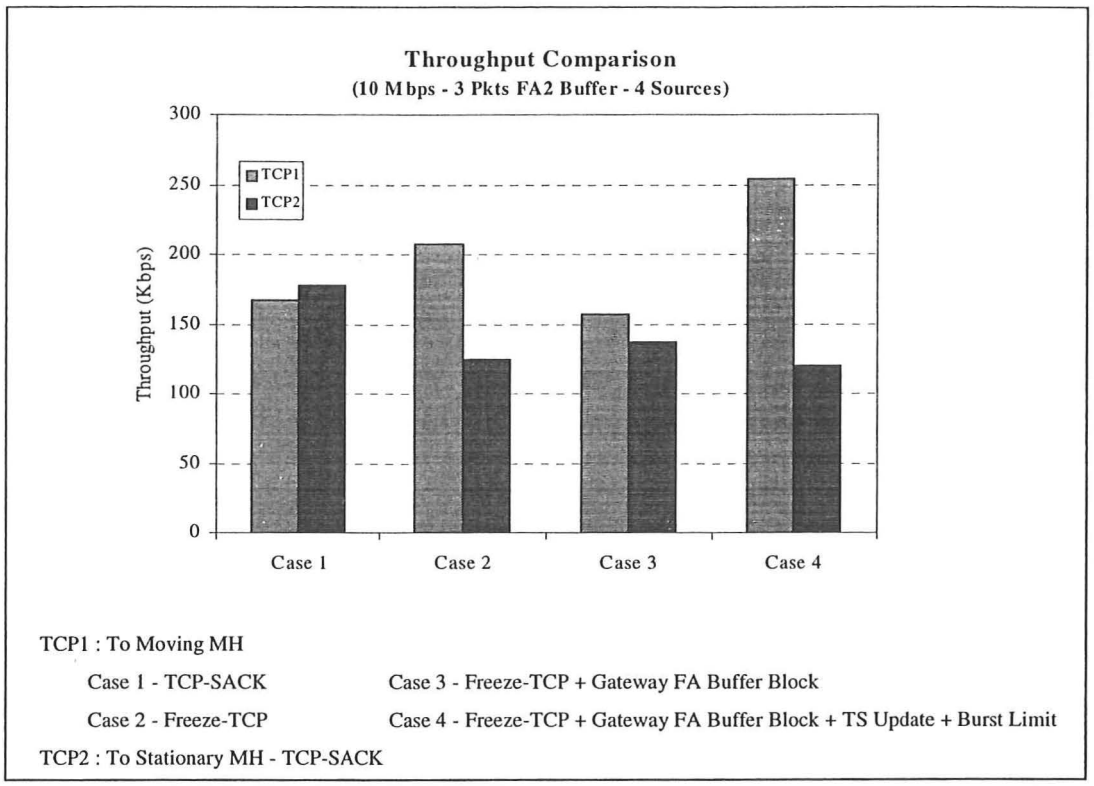

Figure 6.5.2-3: Throughput comparison of TCP-SACK and window freezing mechanisms under load mismatch and uniform link speeds using four background sources.

The figures above showed that the average throughput performance of TCP connections is severely reduced in relation to the case when a 1.5Mbps bottleneck link existed and the BSs have a large capacity mismatch. This is due to the higher congestion observed in the case where all the wired links have bandwidths of 10Mbps. In this scenario, the congestion problem at the BS queues, which interface the 10Mbps wired link and the 2Mbps wireless link is greater. Hence the small buffer with capacity for only three packets causes excessive congestion losses in this scenario.

TCP-SACK

Figure 6.5.2-4 illustrates the congestion window of both TCP connections when TCP-SACK is used in the connections to both the mobile and stationary MHs. Comparison of this figure with Figure 6.4.2-3 shows that the average CWND of the competing

TCP-SACK is now smaller than that in the latter figure. As opposed to Figure 6.4.2-3, in the present scenario, the CWND of the TCP-SACK connection to the stationary MH is always kept below the maximum advertised window (20KB).

Furthermore, the TCP connection to the moving MH is more severely affected by congestion than the corresponding TCP connection in the former link speed scenario, particularly while in the coverage area of the BS with the smaller buffer size. In the present case a timeout affects a connection more severely than in the case of load mismatch in Section 6.4.2 due to the larger congestion level in the current scenario. The competing connection is able to use the resources that were previously used by the connection that timed out and the latter connection has more difficulty in reclaiming resources.

In this scenario, losses at the more congested BS and the occurrence of RTO backoff were seen to lead to idle times even after the mobile leaves this cell. Such fact can be seen at time 80 seconds in Figure 6.4.2-4. The congestion window cannot grow for a large time after the MH entered the less congested cell at this time.

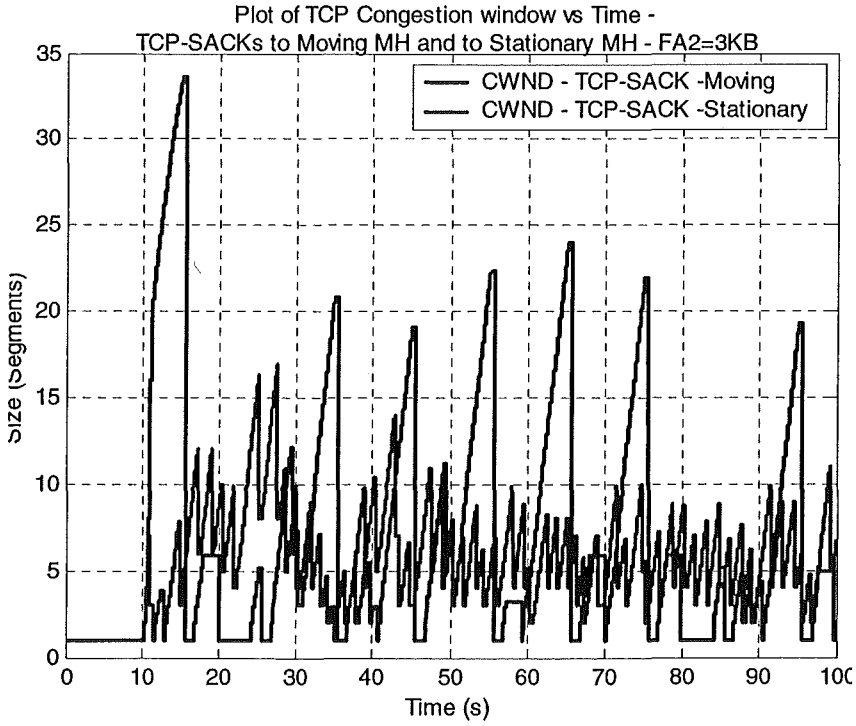

Figure 6.5.2-4: Congestion window of TCP-SACK connections to mobile and stationary mobiles with uniform link speeds – two background sources.

Freeze-TCP

As opposed to the load mismatch scenarios in the previous section, where Freeze-TCP was found to perform well by avoiding timeouts during handoffs, either allowing the sender to continue transmission at the previous rate or requiring only a fast retransmission for loss recovery, in this scenario it was observed that the Freeze-TCP performance was more adversely affected by the burst of data to the new BS. This is shown in the congestion window plot of Figure 6.5.2-5 and the BSs and receiver traces shown in Figure 6.5.2-6 for the case where two background sources are used.

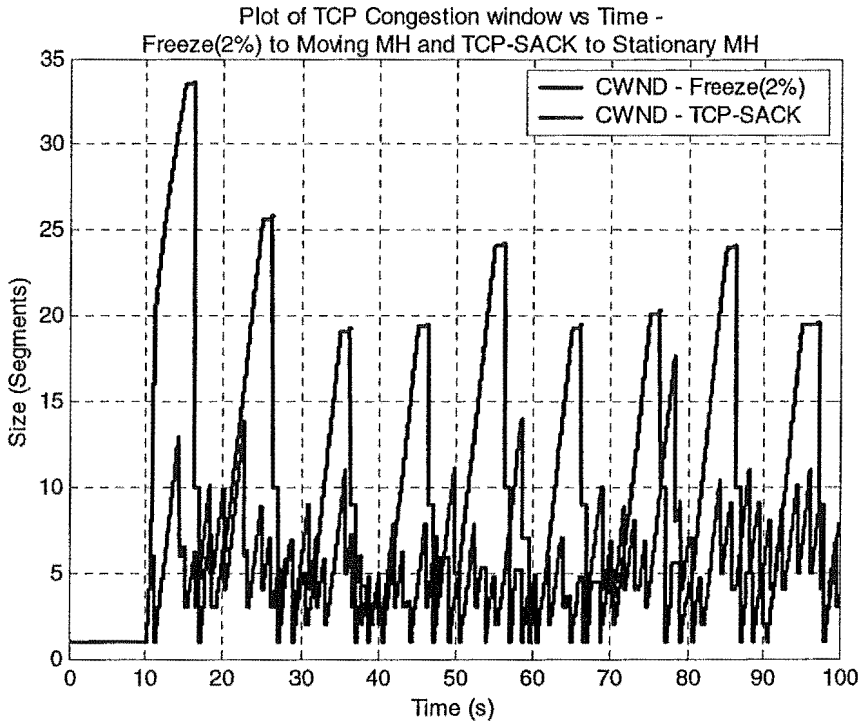

Figure 6.5.2-5: Congestion window of Freeze-TCP to mobile connection and competing TCP-SACK to stationary MH showing timeouts of Freeze-TCP as mobile moves into congested cell – two background sources.

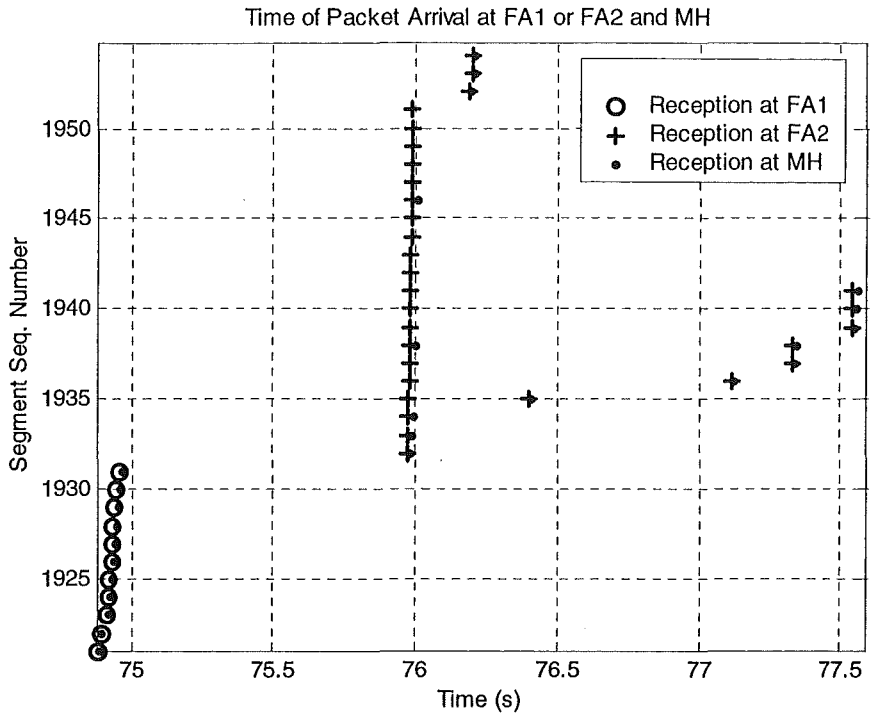

Figure 6.5.2-6: Adverse effect of Freeze-TCP burst following a reconnection into the more congested cell – two background sources.

Figure 6.5.2-6 shows that the burst of data sent to the MH following the handoff at time 75 seconds, when the mobile enters the more congested FA2 coverage area, led to the mobile not receiving several packets sent in this burst. Hence the sender needs to resort to a timeout, which leads to the congestion window reduction shown in Figure 6.5.2-5, to recover from the packet losses.

The increased losses seen by Freeze-TCP during the handoffs to the more congested cell and the large network congestion, which requires a small congestion window size while in this cell, led to the smaller performance improvement of Freeze-TCP over using TCP-SACK in this scenario when compared to the corresponding case where a slower link was used. Although freezing the congestion window does not bring large benefits to Freeze-TCP in this scenario due to the high congestion rate and the excessive bursts of data, freezing the TCP sender was still beneficial. The occurrence of timeouts due to handoffs is therefore avoided and hence exponential RTO backoff is reduced. The use of reconnection notification by the MH is also useful in restarting the sender promptly after the end of the handoff. These factors explain why Freeze-TCP still performed approximately 23% better than if TCP-SACK was used in the

mobile connection in the current scenario for both background traffic levels considered.

Freeze + Gateway Blocking

As in the load mismatch scenarios in the previous section, the window freezing mechanism with buffer blocking led to several timeouts and large idle times due to the bursts from the Gateway FA and the RTO inflation. Note that the buffer blocking scheme now does not bring statistically significant performance improvement in comparison to using TCP-SACK, as opposed to the scenario in Section 6.4.2, where although the former scheme performed worse than Freeze-TCP, gain over using TCP-SACK for the mobile connection was still observed when two background sources were used.

The higher capacity of the pipe between the TCP sender and the Gateway FA due to the increased bandwidth means that more packets are in transit when a disconnection is predicted. Hence there is the potential for further performance degradation due to a larger burst of data from the Gateway FA buffer agent into the congested BS buffer.

Freeze + Gateway Blocking + TS Echo Update + Burst Limit

As in the buffer capacity mismatch scenarios in the previous section, the addition of the burst control and timestamp echo update functions avoid excessive losses and RTO inflation due to the handoffs. Therefore, in this scenario, the proposed buffer block scheme with timestamp echo update and a burst limit of two packets performed on average 32% and 22% better than Freeze-TCP when two and four background sources are used, respectively. In the latter case, the more frequent congestion window reduction due to the increased congestion caused by the heavier background traffic reduces the advantage of the burst control scheme since the congestion window needs to be reduced more often due to the larger congestion, independently of the presence of the burst control or not. This phenomenon was also observed when analysing the results for the load mismatch cases when using the 1.5Mbps wired link.

The congestion window plot of the buffer blocking scheme with timestamp echo updates and burst size control when two background sources are present is shown in Figure 6.5.2-7 and a comparison of typical receiver trace files of TCP-SACK and the window freezing schemes considered in this section is shown in Figure 6.5.2-8. In the former figure, it can be seen that the burst control mechanism still makes possible the recovery from handoffs through a fast retransmission in several occasions. This contrasts to the need for a timeout when all buffered packets are sent in a burst to the congested BS buffer or when Freeze-TCP is used in this scenario.

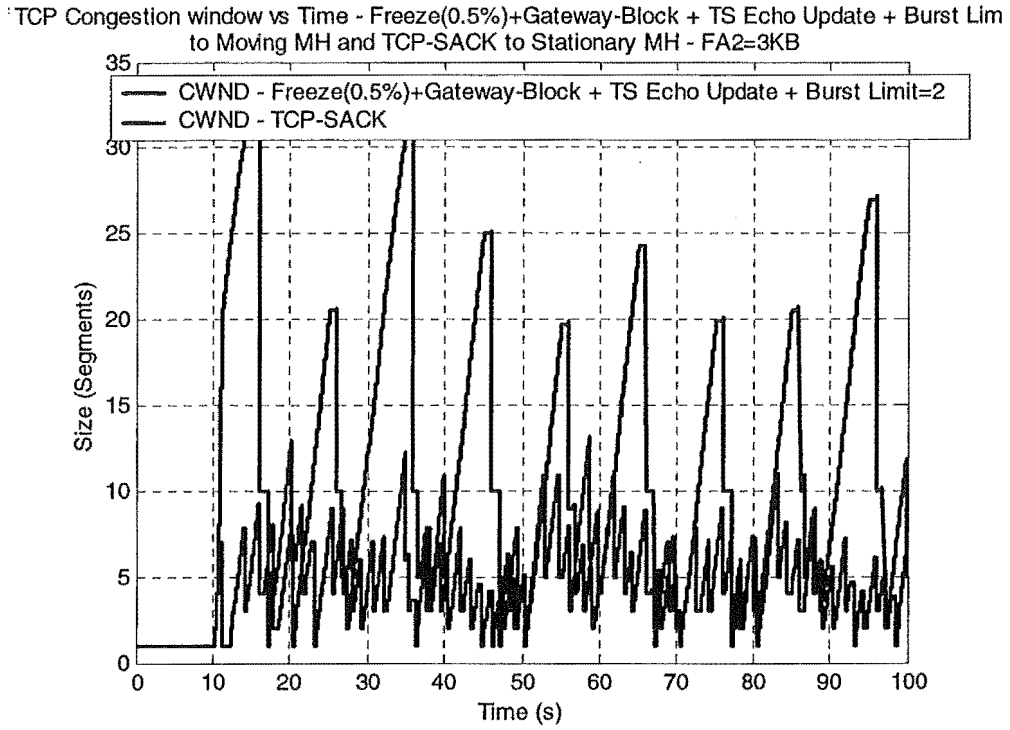

Figure 6.5.2-7: Congestion window of Freeze-TCP with Gateway FA blocking, timestamp echo updates and a burst limit of two packets using two background sources.

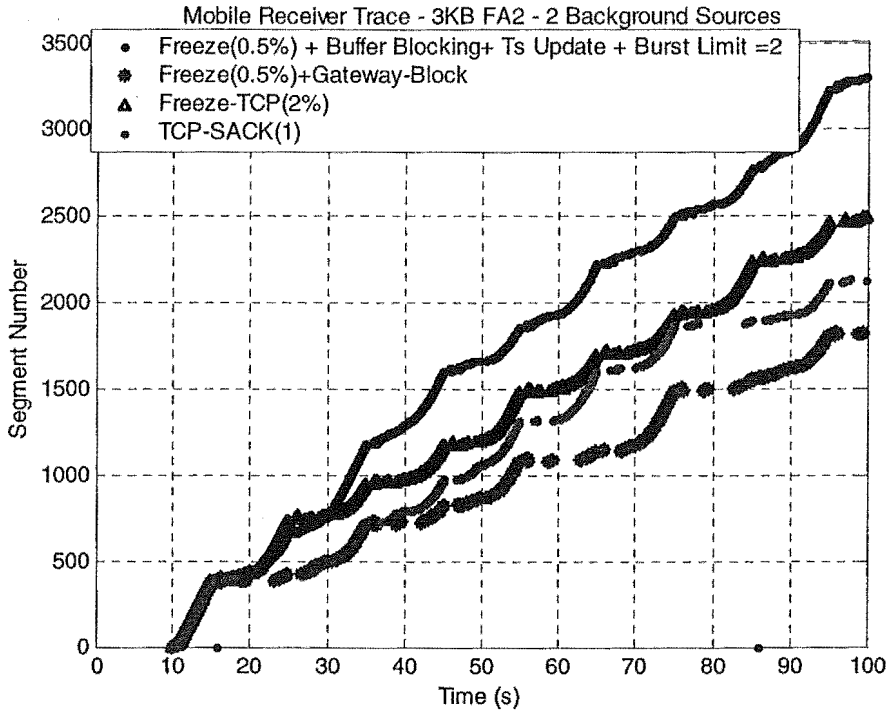

Figure 6.5.2-8: Receiver traces for load mismatch scenario using two background sources and link speeds of 10Mbps throughout the network.

Figure 6.5.2-8 shows that the proposed Freeze-TCP with buffering enhanced with burst limit control and timestamp echo updates further reduced the receiver idle times in comparison to Freeze-TCP and allowed a faster reception of segments. The former effect was seen in previous discussions to be due to the ability of our proposed scheme to avoid RTO inflation and hence allow for faster recovery when a timeout is required. The latter effect was seen to be due to the burst limiting function ability to reduce losses in congested scenarios, hence avoiding congestion window reduction. The figure also shows the inability of the pure buffering scheme in avoiding long receiver idle times due to handoffs in this scenario.

6.5.3. Summary

This section showed that, while no significant improvement was obtained by limiting the burst size sent to the new BS when no heavy congestion existed, the proposed scheme performed significantly better when significant congestion existed in one of

the BSs. This was particularly true when considering a larger link speed mismatch at the BSs.

In this case, all schemes had their throughput reduced due to the larger congestion in the network. In particular, TCP-SACK had more difficulties in reclaiming resources following handoffs. Freeze-TCP was seen to lead to buffer overflow due to inadequate congestion window and timeouts also occurred. On the other hand, using the burst limiting scheme still avoided timeouts during several handoffs. Hence significant improvements of over 20% over Freeze-TCP were observed by using burst limiting. and timestamp echo updates

Since the burst limiting scheme still led to some timeouts during handoffs to congested BSs, the use of the proposed timestamp echo update mechanism was still useful.

6.6. Conclusions

This chapter presented a performance evaluation of the proactive sender-freezing approach considering different schemes to freeze the sender under various scenarios. TCP-SACK was used for comparison purposes.

The improved robustness of the proposed buffering mechanism over the base Freeze-TCP was confirmed through its ability to use lower threshold, and hence freeze the sender later. The proposed buffering mechanism was observed to have a higher independence from the underlying path RTT than the base Freeze-TCP.

The simulations in this chapter showed that TCP-SACK is severely penalised by handoffs due to the need to reduce its transmission rate even when the new BS has enough capacity to support the previous rate. Furthermore, the difficulty in reclaiming resources after a timeout in heavily congested scenarios where a competing stationary TCP connection existed was seen to adversely affect the TCP-SACK performance further.

The window freezing approach avoided these problems by reducing the occurrence of timeouts to restart transmission. The avoidance of unnecessary transmission rate reduction led to significant throughput improvement over TCP-SACK of up to approximately 80% in the scenarios tested.

The use of buffer blocking, however, was seen to increase the likelihood of buffer overflow due to the lower delay in the path to the congested BS. The link speed mismatch was the other important factor in the level of losses observed and in the capacity of the BS buffers to handle bursts from either the TCP sender or the Gateway FA following a reconnection detection.

The proposed timestamp echo update mechanism was seen to bring significant performance enhancements in cases where a timeout is required following a reconnection where multiple packets are lost during the disconnection period. This mechanism avoided the unnecessary inclusion of a disconnection period in the RTT calculation and hence avoided new ACKs from inflating the RTO. Hence faster recovery at the sender was possible than when the mechanism was not used. This was especially true for larger disconnection periods.

This timestamp echo updates scheme was of particular importance when a proactive buffer scheme was used and the burst of data sent following a reconnection caused buffer overflow. Hence our proposed timestamp echo update scheme avoided the use of the old timestamps from buffered packets in the RTT updates.

The proposed burst limiting scheme was seen to significantly reduce the problem of buffer overflow due to an excessive burst of data following a reconnection by limiting the amount of data that could be sent to the new BS in a single burst. Hence the combination of buffer blocking with timestamp echo updates and the burst limit scheme was seen to provide improvements of up to approximately 30% over the base Freeze-TCP in cases of large congestion and link speed mismatch.

Chapter 7. Conclusions and Future Work

7.1. Conclusions

Using TCP over wireless presents many challenges due to its assumption that losses are caused by congestion in the network and its inability to distinguish different causes of losses. Important related work in the area of improving TCP performance to deal with both the high loss rate and the problem of mobility were reviewed in this thesis.

Mechanisms based on the proactive window freezing of the Freeze-TCP concept were discussed in detail. This work extends previous window freezing simulation works in that we also simulate the effect of the window freezing on a competing TCP connection and with different background traffic levels.

The proactive end-to-end window freezing mechanism of Freeze-TCP was implemented in a simulator and its performance studied under scenarios of frequent disconnections. TCP-Reno with SACK was used as the baseline for comparisons due to the suitability of SACK for wireless environments according to several previous researches and wide deployment in the current Internet.

Although the proactive window freezing mechanism's ability to avoid unnecessary congestion window reduction and excessive idle times caused by mobility allowed significant throughput performance enhancement over TCP-SACK under several different scenarios, shortcomings of this scheme were observed through simulations. In particular, the problem of determining a proper fixed power warning threshold was observed. This was due to the difficulty in avoiding freezing the sender too soon while also avoiding the loss of in-flight data, given the fact that packets experience widely varying delays due to dynamically changing network conditions.

A buffering scheme triggered by a disconnection prediction at a point close to the MH was proposed to reduce this problem. Such proactive buffering avoids the need to snoop the TCP header by allowing the buffer agent to snoop the IP header instead of the TCP header. Hence certain types of IP encryption can still be used together with the Freeze-TCP concept. It also triggers buffering only when the MH predicts a disconnection.

Mechanisms were proposed to further reduce the occurrence of losses when a proactive disconnection prediction mechanism is used and to allow faster recovery when a retransmission timeout is still required for recovery. Specifically, the buffer agent proposed was modified to limit the amount of data sent in a burst to the MH following reconnection detection.

The other modification proposed in the thesis relates to the attempt to allow faster recovery when a timeout is required due to multiple losses during a disconnection. It attempts to avoid unnecessary sender idle times by avoiding RTT estimates being inflated by the disconnection period. No changes to the fixed host sender or other wired network nodes are needed for this. Hence compatibility with the existing infrastructure is maintained.

The simulation results showed that the proposed buffering mechanism was capable of substantially increasing the independence from the wired link delay and increasing the throughput performance in comparison with base Freeze-TCP. This was possible by allowing the use of a smaller threshold than that required by the end-to-end Freeze-TCP in cases where no heavy congestion existed in the BSs. Both the Freeze-TCP and Freeze-TCP with buffering and timestamp echo updates had significant performance enhancement over TCP-SACK when we tested different thresholds under low congestion.

The simulations of the use of Freeze-TCP with the proposed timestamp echo updates showed improvements over base Freeze-TCP when a timeout was required due to the use of a threshold not large enough to avoid the need for a timeout. This was seen to be particularly beneficial when a larger disconnection period existed due to the further

increase in the RTT estimate containing the old timestamp. An improvement of approximately 19% was obtained in this case.

The use of competing TCPs under different congestion levels and buffer capacities showed that base Freeze-TCP avoided the unfairness observed against TCP-SACK, where the latter was unable to restart significant transmission rate growth following a timeout. Furthermore, results showed that when enough buffer space was available and the lower level of background was used, the competing TCP-SACK was not affected by Freeze-TCP and in fact had an improvement of up to 10% in relation to the case when TCP-SACK was used in the mobile connection.

Our results showed that even when the congestion window used by the base Freeze-TCP was inappropriate for the new cell conditions, it was still able to outperform TCP-SACK. However, the link speed was seen to significantly impact on the ability of base Freeze-TCP in avoiding losses due to inappropriate window size. Even when heavy congestion existed, Freeze-TCP was still able to perform over 20% better than TCP-SACK in our tests.

The use of background traffic and a competing TCP connection was particularly important when evaluating the buffering scheme's performance. The results showed that while this mechanism is very effective in low congestion scenarios, it leads to increased penalty when the BSs are congested. Using the proactive buffering scheme alone was therefore seen to lead to worse performance than Freeze-TCP but better performance than TCP-SACK in the mobile connection in most cases tested. This contrasts with previous research in [Onoe et al 2001] where window freezing mechanisms were simulated without the use of a competing TCP. In that study no performance degradation was observed due to the use of proactive buffering when compared to their end-to-end scheme implementation.

The use of the proposed timestamp echo update was particularly beneficial in such scenarios of high BS congestion as it avoided further idle times due to the use of old timestamps in new ACKs of buffered packets. The use of the burst limiting scheme was seen to reduce the performance degradation caused by excessive bursts to the congested BS. Hence it achieved comparable performance to Freeze-TCP when the

latter sent data through a bottleneck link. On the other hand, when all links had the same speed, the use of burst control was seen to provide a significant performance improvement of between approximately 22% and 32% over Freeze-TCP depending on whether high or low background levels were used, respectively. Coupling the buffer blocking with a burst limit and the timestamp echo update mechanism was observed to be useful for increased robustness.

7.2. Future Work

As noted in the literature review, the task of adapting TCP for enhanced performance against the wireless link impairments and mobility requires modifications targeted at solving different issues relating to TCP performance degradation. The modifications proposed in this thesis are targeted at the problem of mobile disconnection. Other solutions should be used in conjunction with those investigated in this work for improved TCP performance in a real wireless environment.

One of the objectives of this thesis was to improve Freeze-TCP while still allowing the use of IPSec encryption by avoiding the need for snooping the TCP header. Hence a solution that would be coupled to the proactive window freezing schemes investigated should attempt to maintain this feature. One such approach is the use of link layer acknowledgements, such as the scheme of Delayed Duplicate Acknowledgements [Vaidya et al 1999].

Although the modifications proposed were tested under several different scenarios, it would be interesting to test such schemes under other conditions also. In particular, the effect on the unmodified TCP could be further studied under more complex scenarios.

It is observed that the mobile controlled window freezing mechanism's ability to freeze the sender and avoid packet losses is dependent on the mobile's speed and the rate of signal fading. Therefore, controlled conditions were used to trigger the types of conditions that are of interest for the situations studied and to allow a meaningful interpretation of the behaviours observed. Although this approach was useful for our

aim of evaluating the schemes under controlled conditions, it is emphasised that more sophisticated channel models are needed for a future detailed study of the ability of this mechanism to detect a disconnection in real environments due to its dependence on the mobile speed and speed of signal decay. It is hence important to use a real testbed when further testing this scheme, possibly with the aid of simulation tools as previously described for a better understanding of results.

It is clear that more work is required for the determination of a proper threshold value to be used in different situations, such as different MH speeds and fading scenarios. The development of dynamic detection mechanisms is an area that needs attention in future work. This would allow the MH to efficiently detect an incoming disconnection in real dynamic scenarios.

The simulation work in this thesis focused on the environment of cellular networks. However, the proposed end-to-end timestamp echo update mechanism may be particularly useful in ad-hoc networks. In such networks no fixed infrastructure is available to mitigate the problems of mobility and hence an end-to-end solution, or one that does not depend on a central node, is required. Therefore an interesting issue to be studied in the future is the evaluation of this mechanism in such networks. The long disconnection time caused by host mobility and route discovery may yield greater enhancement when using the mechanism in such networks compared to the case of cellular networks.

It would also be interesting to evaluate the performance of the proactive buffer blocking mechanism in ad-hoc networks. This may be beneficial in that the buffer agent may be implemented in the MHs and hence it stops transmission when a disconnection is predicted, therefore saving transmission power on the transmission of packets likely to be lost. This would come at the cost of increased buffering requirements at the MH to store data for later redirection to the disconnecting MH. Methods for the latter MH to warn the former of its new location and reconnection events must be studied.

References

[Allman and Falk 1999] - Mark Allman and Aaron Falk, "On the Effective Evaluation of TCP," *ACM Computer Communication Review*, vol. 29, 1999.

[Bakre 1996] - A. Bakre, "Design and Implementation of Indirect Protocols for Mobile Wireless Environments", PhD Thesis, New Brunswick: The State Univesity of New Jersey, 1996.

[Bakre and Badrinath 1995] – A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Host", Proceedings of the 15th International Conference on Distributed Computing Systems, pp. 136-143, Vancouver, Canada, 1995.

[Bakre and Badrinath 1997] - Ajay V. Bakre and B. R. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Transactions on Computers*, vol. 46, pp. 260 - 278, 1997.

[Bakshi et al 1997] - B. S. Bakshi, P. Krishna, N. H. Vaidya and D. K. Pradhan, "Improving Performance of TCP over Wireless Networks," Proceedings of the 17th International Conference on Distributed Computing Systems, Baltimore, MD, pp. 365-373, 1997.

[Balakrishnan et al 1995] - H.Balakrishnan, S. Seshan and R. H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," *ACM Wireless Networks*, vol. 1, 1995.

[Balakrishnan et al 1997] - Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan and Randy H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 756-769, 1997.

[Balakrishnan 1998] – Hari Balakrishna, "Challenges to reliable Data Transport over Heterogeneous Wireless Networks," in *Department of Electrical Engineering and Computer Sciences, Computer Science Division*. Berkeley: University of California at Berkeley, 1998.

[Blanton 2000] - Ethan Blanton, "dup ACK behaviour with SACK," http://www.isi.edu/nsnam/archive/ns-users/webarch/2000/msg03497.html, 2000.

[Braden 1989] - R. Braden, "Requirements for Internet Hosts - Communication Layers," RFC 1122, http://www.ietf.org/rfc/rfc1122.txt?number=1122, 1989.

[Brewer et al 1998] - Eric A. Brewer, Randy H. Katz, Yatin Chawathe, Steven D. Gribble, Todd Hodes, Giao Nguyen, Mark Stemm, Tom Henderson, Elan Amir, Hari Balakrishnan, Armando Fox, Venkata N. Padmanabhan and Srinivasan Seshan, "A Network Architecture for Heterogeneous Mobile Computing," in *IEEE Personal Communications*, vol. 5, 1998, pp. 8-24.

[Brown and Singh 1997] - Kevin Brown and Suresh Singh, "M-TCP: TCP for Mobile Cellular Networks," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 27, pp. 19-43, 1997.

[Cáceres and Iftode 1994] - Ramón Cáceres and Liviu Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 850-857.

[Cáceres and Padmanabhan 1998] - Ramón Cáceres and Venkata N. Padmanabhan, "Fast and Scalable Wireless Handoffs in Support of Mobile Internet Audio," ACM Journal on Mobile Networks and Applications, vol. 3, 1998.

[Campbell et al 2000] - Andrew T. Campbell, Javier Gomez, Sanghyo Kim, András G. Valkó, Chieh-Yih Wan and Zoltán R. Turányi, "Design, Implementation, and Evaluation of Cellular IP," in *IEEE Personal Communications*, vol. 7, 2000, pp. 42-49.

[Chan et al 1997] - Aldar C.-F. Chan, Danny H. K. Tsang and Sanjay Gupta, "Impacts of Handoff on TCP Performance in Mobile Wireless Computing," presented at Personal Wireless Communications, 1997 IEEE International Conference on, Mumbai, India, 1997.

[Chen and Lee 2000] - Wen-Tsuen Chen and Jyh-Shin Lee, "Some Mechanisms to Improve TCP/IP Performance over Wireless and Mobile Computing Environment, "Proceedings 7th International Conference on Parallel and Distributed System, 2000", pp. 437-444, 2000.

[Comer 1991] - Douglas E. Comer, *Internetworking with TCP/IP*, vol. 1, 2nd ed. Englewood Cliffs., N.J.: Prentice Hall, 1991.

[de Silva 2001] - Prasan de Silva, "Micro-Mobile IP (uMIP): Mobility Management in IP Based Cellular Networks," in *Electrical and Computer Engineering*. Christcurch, New Zealand: University of Canterbury, 2001.

[DeSimone et al 1993] – A. DeSimone, M. C. Chuah and O. C. Yue, "Throughput Performance of Transport Layer Protocols over Wireless LANs," in Proceedings IEEE GOBECOM'93, 1993.

[Duchamp and Reynolds 1992] – D. Duchamp and N. Reynolds, "Measured Performance of a Wireless LAN," 17th Conference on Local Computer Networks, Minneapolis, MN, pp. 494-499, 1992.

[Elaoud and Ramanathan 2000] - Moncef Elaoud and Parameswaran Ramanathan, "TCP-SMART: A Technique for Improving TCP Performance in a Spotty Wide Band Environment," Proceedings of IEEE ICC'2000, pp. 1783-1787, 2000.

[Fall and Floyd 1996] - Kevin Fall and Sally Floyd, "Simulation-based Comparison of Tahoe, Reno, and SACK TCP," *Computer Communication Review*, vol. 26, 1996.

[Fall and Varadhan 2000] - Kevin Fall and Kannan Varadhan, "The ns Manual," http://www.isi.edu/nsnam/ns/index.html, 2000.

[Floyd 1994] – S. Floyd, "TCP and Explicit Congestion Notification", ACM Computer Communications Review, vol. 24, 1994.

[Floyd 2001] – S. Floyd, "Sally Floyd: Questions," http://www.icir.org/floyd/questions.html, 2001.

[Goff et al 2000] - Tom Goff, James Moronski, D. S. Phatak, Vipul Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," presented at IEEE INFOCOM 2000, 2000.

[Gustafsson et al 2001] - Eva Gustafsson, Annika Jonsson and Charles E. Perkins "Mobile IPv4 Regional Registration," http://search.ietf.org/internet-drafts/draft-ietf-mobileip-reg-tunnel-06.txt, 2001.

[Jacobson 1992] - V. Jacobson, R. Braden and D. Borman, "TCP Extensions for High Performance," RFC 1323, http://www.ietf.org/rfc/rfc1323.txt?number=1323, 1992.

[Kent and Atkinson 1998] - S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," , RFC 2041 (Standards Track), http://www.ietf.org/rfc/rfc2041.txt?number=2041, 1998.

[Kuhlberg 2000] - Panu Kuhlberg, "Effect of Delay and Packet Drops on TCP-based Wireless Communication," in *Department of Computer Science*. Helsinki: University of Helsinki, 2000.

[Ludwig 2000] - Reiner Ludwig, "Eliminating Inneficient Cross-Layer Interactions in Wireless Networking," PhD Thesis, Aachen: Aachen University of Technology, 2000.

[Ludwig and Katz 2000] - Reiner Ludwig and Randy H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions," *ACM Computer Communication Review*, vol. 30, pp. 30-36, 2000.

[Mathis et al 1996] - M. Mathis, J. Mahdani, S. Floyd, and A. Romanow, "TCP

Selective Acknowledgement Options," RFC 2018 (Standards Track), http://www.ietf.org/rfc/rfc2018.txt?number=2018, 1996.

[Montenegro et al 2000] – "Long Thin Networks," RFC 2757, http://www.ietf.org/rfc/rfc2757.txt?number=2757, 2000.

[Onoe et al 2001] - Yuko Onoe, Yukio Atsumi, Fumiaki Sato and Tadanori Mizuno, "An Efficient TCP/IP Control Scheme for Next-Generation MobileIP Communication Networks," *IEICE Transactions on Communications*, vol. E84-B, pp. 863-872, 2001.

[Perkins 1996] - C. Perkins, "IP Mobility Support," RFC 2002, Standards Track, http://www.ietf.org/rfc/rfc2002.txt?number=2002, 1996.

[Perkins and Johnson 2001] – Charles Perkins and David B. Johnson, "Route Optimization in Mobile IP, http://search.ietf.org/internet-drafts/draft-ietf-mobileip-optim-11.txt, 2001

[Perkins and Bhagwat 1994] - Charles E. Perkins and Pravin Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computing," *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994.

[Rappaport 1996] - Theodore S. Rappaport, "Wireless Communications: Principles and Practice", Upper Saddle River, New Jersey: Prentice Hall PTR, 1996.

[Sarolahti 2000] - Pasi Sarolathi, "Performance Analysis of TCP Enhancements for Congested Reliable Wireless Links," Master's Thesis, in *Department of Computer Science*. Helsinki: University of Helsinki, 2000.

[Seshan 1995] – S. Shean, "Low Latency Handoff for Cellular Data Networks," PhD Thesis, Berkeley: University of California at Berleley, 1995.

[Seshan et al 1996] - Srinivasan Seshan, Hari Balakrishnan and Randy H. Katz,

"Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience," *Kluwer International Journal on Wireless Communication Systems*, 1996.

[Stallings 2000] - Wiliam Stallings, *Data & computer Communications*, Sixth ed. Upper Saddle River, New Jersey: Prentice Hall, 2000.

[Stevens 1994] - W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, vol. 1. Reading, Mass.: Addison-Wesley Pub. Co, 1994.

[Vaidya 1999] – "TCP for Windows and Mobile Hosts", Texas A&M University, http://www.cs.tamu.edu/faculty/vaidya/, 1999.

[Vaidya et al 1999] - Nitin Vaidya, Miten Mehta, Charles Perkins and Gabriel Montenegro, "Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless," Technical Report 99-003, Computer Science Dept., Texas A&M University, Feb., 1999.

[Widmer 2001] - J. Widmer, "Extensions to the ns Network Simulator," http://www.icsi.berkeley.edu/~widmer/mnav/ns-extension/, 2001.

[Wright and Stevens 1995] - G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*, vol. 2. Reading, Mass.: Addison-Wesley Pub. Co., 1995.

[Xylomenos et al 2001] - G. Xylomenos, G. C. Polyzos, P. Mähönen and M. Saaranen, "TCP Performance Issues Over Wireless Links," in *IEEE Communications Magazine*, vol. 39, 2001, pp. 52-58.

Appendix A: Full Results of Competing TCPs Tests

This appendix presents the complete average simulation results of the performance metrics considered for each scenario discussed in sections 6.4 and 6.5. The percentage improvement comparisons among the schemes considered is also shown for each scenario.

The Troughput Results tables show the mean throughput and its standard deviation (STD) for each scheme in a given scenario.

The Throughput Comparison tables show the percentage improvement comparison between different schemes in each scenario.

The last table of each scenario shows the mean values of the throughput (tput), goodput, number of timeouts (Timeouts) and number of retransmissions during the simulation duration (rexmit).

1.5Mbps – 20 Packet buffers – 2 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	307.66	3.96	508.21	8.91
Freeze	446.52	8.64	542.50	11.70
Block	470.04	9.30	517.84	10.91
Block + Update	481.88	10.16	474.68	9.47
Block + Update + Limit	470.41	5.50	494.32	11.14

Table A-1: Throughput results - 1.5Mbps wired link; 20 packet buffers; two background sources

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	45.1	6.7
SACK	Block	52.8	*
SACK	Block + Update	56.6	-6.6
SACK	Block + Update + Limit	52.9	-2.7
Freeze	Block	*	*
Freeze	Block + Update	7.9	-12.5
Freeze	Block + Update + Limit	5.4	-8.9
Block	Block + Update	*	-8.3
Block	Block + Update + Limit	*	*
Block + Update	Block + Update + Limit	*	*

* Not statistically significant

Table A- 2: Throughput comparisons - 1.5Mbps wired link; 20 packet buffers; two background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	307.66	0.93	17.80	264,600	508.21	0.98	2.60	90,000
Freeze	446.52	0.99	0.70	23,900	542.50	0.99	0.30	30,200
Block	470.04	0.99	4.33	62,583	517.84	0.99	1.08	49,500
Block + Update	481.88	0.99	3.29	54,571	474.68	0.99	2.14	68,857
Block + Update + Limit	470.41	0.99	2.82	55,588	494.32	0.99	1.88	59,941

Table A-3: Overall Average Results - 1.5Mbps wired link; 20 packet buffers; two background sources.

1.5Mbps – 20 Packet buffers – 4 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	285.54	1.12	445.63	10.21
Freeze	414.62	8.54	412.91	8.31
Block	432.52	8.44	393.72	9.46
Block + Update	417.89	14.82	394.88	7.64

Table A-4: Throughput results - 1.5Mbps wired link; 20 packet buffers; four background sources.

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	45.2	-7.3
SACK	Block	51.5	-11.6
SACK	Block + Update	46.4	-11.4
Freeze	Block	*	*
Freeze	Block + Update	*	*
Block	Block + Update	*	*

* Not statistically significant

Table A-5: Throughput comparisons - 1.5Mbps wired link; 20 packet buffers; four background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	285.54	0.93	17.95	244,700	445.63	0.97	5.00	111,900
Freeze	414.62	0.99	0.71	33,285	412.91	0.98	3.29	89,857
Block	432.52	0.98	3.65	66,434	393.72	0.97	4.52	103,826
Block + Update	417.89	0.98	3.86	72,285	394.88	0.97	4.71	110,285

Table A-6: Overall Average Results - 1.5Mbps wired link; 20 packet buffers; four background sources.

1.5Mbps – 3 Packet buffers – 2 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	245.83	9.89	372.87	7.83
Freeze	442.03	17.29	305.26	6.68
Block	292.59	10.44	343.08	7.29
Block + Update	352.16	3.87	308.29	6.27
Block + Update + Limit	436.90	6.64	280.78	6.15

Table A-7: Throughput results - 1.5Mbps wired link; 3 packet buffers; two background sources.

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	79.8	-18.1
SACK	Block	19.0	-8.0
SACK	Block + Update	43.3	-17.3
SACK	Block + Update + Limit	77.7	-24.7
Freeze	Block	-33.8	12.4
Freeze	Block + Update	-20.3	*
Freeze	Block + Update + Limit	*	-8.0
Block	Block + Update	20.4	-10.1
Block	Block + Update + Limit	49.3	-18.2
Block + Update	Block + Update + Limit	24.1	-8.9

* Not statistically significant

Table A-8: Throughput comparisons - 1.5Mbps wired link; 3 packet buffers; two background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	245.83	0.93	20.69	215,384	372.87	0.98	3.46	92,538
Freeze	442.03	0.99	0.69	24,692	305.26	0.97	3.08	80,538
Block	292.59	0.95	11.08	179,230	343.08	0.97	3.46	88,846
Block + Update	352.16	0.95	10.10	181,100	308.29	0.97	4.50	100,100
Block + Update + Limit	436.90	0.98	1.79	100,785	280.78	0.97	4.29	98,357

Table A-9: Overall Average Results - 1.5Mbps wired link; 3 packet buffers; two background sources.

1.5Mbps – 3 Packet buffers – 4 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	244.05	5.58	294.88	6.88
Freeze	415.32	5.55	232.79	5.37
Block	255.58	3.62	260.26	6.04
Block + Update	313.10	5.10	253.32	5.90
Block + Update + Limit	433.68	4.29	218.38	4.82

Table A-10: Throughput results - 1.5Mbps wired link; 3 packet buffers; four background sources.

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	70.2	-21.1
SACK	Block	*	-11.7
SACK	Block + Update	28.3	-14.1
SACK	Block + Update + Limit	77.7	-25.9
Freeze	Block	-38.5	11.8
Freeze	Block + Update	-24.6	8.8
Freeze	Block + Update + Limit	4.4	*
Block	Block + Update	22.5	*
Block	Block + Update + Limit	69.7	-16.1
Block + Update	Block + Update + Limit	38.5	-13.8

* Not statistically significant

Table A-11: Throughput comparisons - 1.5Mbps wired link; 3 packet buffers; four background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	244.05	0.92	21.79	218,263	294.88	0.96	8.00	112,000
Freeze	415.32	0.99	1.30	30,700	232.79	0.96	8.25	98,400
Block	255.58	0.94	11.38	178,571	260.26	0.96	8.19	109,190
Block + Update	313.10	0.95	12.20	192,350	253.32	0.96	8.40	113,850
Block + Update + Limit	433.68	0.98	1.20	99,733	218.38	0.95	8.60	113,333

Table A-12: Overall Average Results - 1.5Mbps wired link; 3 packet buffers; four background sources.

10Mbps – 20 Packet buffers – 2 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	292.11	12.31	482.91	9.38
Freeze	455.90	6.00	538.26	10.35
Block	464.25	7.49	495.71	11.12
Block + Update + Limit	447.87	8.13	502.91	11.75

Table A-13: Throughput results - 10Mbps wired link; 20 packet buffers; two background sources.

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	56.1	11.5
SACK	Block	58.9	*
SACK	Block + Update + Limit	53.3	*
Freeze	Block	*	-7.9
Freeze	Block + Update + Limit	*	-6.6
Block	Block + Update + Limit	*	*

* Not statistically significant

Table 14: Throughput comparisons - 10Mbps wired link; 20 packet buffers; two background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	292.11	0.92	21.43	276,571	482.91	0.98	4.86	111,571
Freeze	455.90	0.99	0.63	26,250	538.26	0.99	0.38	34,750
Block	464.25	0.99	2.81	61,000	495.71	0.99	2.00	67,625
Block + Update + Limit	447.87	0.98	4.53	74,842	502.91	0.99	2.11	62,052

Table A-15: Overall Average Results - 10Mbps wired link; 20 packet buffers; two background sources.

10Mbps – 3 Packet buffers – 2 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	177.09	7.91	182.86	4.08
Freeze	217.98	4.09	164.55	3.74
Block	163.38	8.91	171.29	2.39
Block + Update + Limit	289.13	5.83	158.95	3.64

Table A-16: Throughput results - 10Mbps wired link; 3 packet buffers; two background sources.

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	23.1	-10.0
SACK	Block	*	-6.3
SACK	Block + Update + Limit	63.3	-13.1
Freeze	Block	-25.0	*
Freeze	Block + Update + Limit	32.6	*
Block	Block + Update + Limit	77.0	-7.2

* Not statistically significant

Table A-17: Throughput comparisons - 10Mbps wired link; 3 packet buffers; two background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	177.09	0.91	27.36	176,928	182.86	0.95	7.21	93,357
Freeze	217.98	0.93	19.30	183,700	164.55	0.95	9.15	96,750
Block	163.38	0.93	14.60	135,600	171.29	0.95	10.80	100,800
Block + Update + Limit	289.13	0.96	7.53	138,000	158.95	0.94	10.80	102,333

Table A-18: Overall Average Results - 10Mbps wired link; 3 packet buffers; two background sources.

10Mbps – 3 Packet buffers – 4 Sources

Throughput Results

TCP1	To Mobile MH (TCP1)		To Stationary MH (TCP-SACK)	
	Mean	STD	Mean	STD
SACK	167.71	6.23	178.49	3.33
Freeze	207.30	2.32	124.81	2.98
Block	157.47	3.07	137.53	3.30
Block + Update + Limit	253.91	5.40	119.55	2.88

Table A-19: Throughput results - 10Mbps wired link; 3 packet buffers; four background sources.

Throughput Comparisons

Base Scheme	Modified Scheme	% Improvement Over Base Scheme	
		TCP1	Stationary TCP-SACK
SACK	Freeze	23.6	-30.1
SACK	Block	*	-22.9
SACK	Block + Update + Limit	51.4	-33.0
Freeze	Block	-24.0	10.2
Freeze	Block + Update + Limit	22.5	*
Block	Block + Update + Limit	61.2	-13.1

* Not statistically significant

Table A-20: Throughput comparisons - 10Mbps wired link; 3 packet buffers; four background sources.

Overall Results

TCP1	To Mobile MH (TCP1)				To Stationary MH (TCP-SACK)			
	Throughput	Goodput	Timeouts	Rexmit	Throughput	Goodput	Timeouts	Rexmit
SACK	167.71	0.91	29.00	175,000	178.49	0.95	7.29	95,714
Freeze	207.30	0.93	20.18	181,705	124.81	0.93	16.59	105,147
Block	157.47	0.93	15.58	137,000	137.53	0.93	14.42	104,666
Block + Update + Limit	253.91	0.95	12.79	152,041	119.55	0.92	19.38	109,458

Table A-21: Overall Average Results - 10Mbps wired link; 3 packet buffers; four background sources.

Appendix B: Tests on Means of Normal Distributions, Variance Unknown

We assume that there are two normal populations with unknown means μ_1 and μ_2 and unknown variances σ_1^2 and σ_2^2. We wish to test the hypothesis that the two means are equal; that is,

H_0: $\mu_1 = \mu_2$

H_1: $\mu_1 \neq \mu_2$

If we cannot reasonably assume that $\sigma_1^2 = \sigma_2^2$ the test procedure must take this into account. Also, as σ^2 is unknown, it must be estimated by S^2. Therefore we have the test statistics

$$t_0 = \frac{\left(\overline{x_1} - \overline{x_2}\right)}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

and the number of degrees of freedom for t are

$$\nu = \frac{\left(w_1 + w_2\right)^2}{\frac{w_1^2}{n_1 + 1} + \frac{w_2^2}{n_2 + 1}} - 2$$

where

$$w_1 = S_1^2 / n_1$$

and

$$w_2 = S_2^2 / n_2$$

The null hypothesis H_0: $\mu_1 = \mu_2$ is rejected if $|t_0| > t_{\alpha/2,\nu-1}$, where $t_{\alpha/2,\nu-1}$ denotes the upper $\alpha/2$ percentage point of the t distribution with ν-1 degrees of freedom.